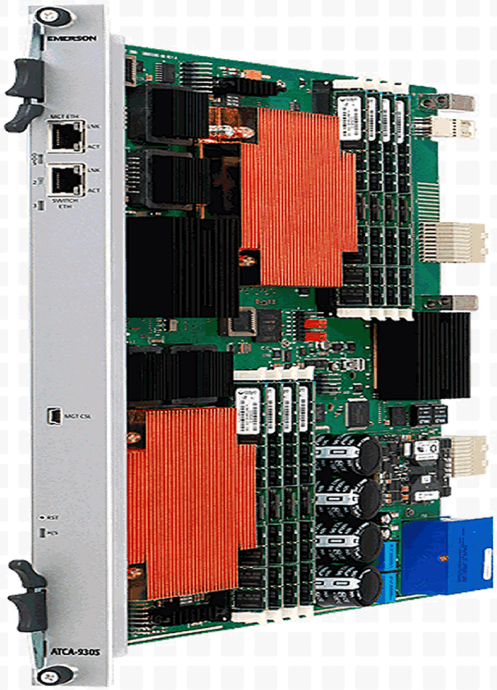
User's Manual
from Emerson Network Power™
Embedded Computing

ATCA-9305: ATCA® Blade with Dual Cavium Processors

April 2009



EMERSON™
Network Power

The information in this manual has been checked and is believed to be accurate and reliable. HOWEVER, NO RESPONSIBILITY IS ASSUMED BY EMERSON NETWORK POWER, EMBEDDED COMPUTING FOR ITS USE OR FOR ANY INACCURACIES. Specifications are subject to change without notice. EMERSON DOES NOT ASSUME ANY LIABILITY ARISING OUT OF USE OR OTHER APPLICATION OF ANY PRODUCT, CIRCUIT, OR PROGRAM DESCRIBED HEREIN. This document does not convey any license under Emerson patents or the rights of others.

Revision Level:	Principal Changes:	Date:
10009109-00	Original release	January 2009
10009109-01	Added “GR-1089-CORE Standard” on page -i Updated “Product Certification” on page 1-4	April 2009

Regulatory Agency Warnings & Notices

The Emerson ATCA-9305 meets the requirements set forth by the Federal Communications Commission (FCC) in Title 47 of the Code of Federal Regulations. The following information is provided as required by this agency.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RULES AND REGULATIONS – PART 15

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Caution: **Making changes or modifications to the ATCA-9305 hardware without the explicit consent of Emerson Network Power could invalidate the user's authority to operate this equipment.**

EMC COMPLIANCE

The electromagnetic compatibility (EMC) tests used an ATCA-9305 model that includes a front panel assembly from Emerson Network Power.

Caution: **For applications where the ATCA-9305 is provided without a front panel, or where the front panel has been removed, your system chassis/enclosure must provide the required electromagnetic interference (EMI) shielding to maintain EMC compliance.**

GR-1089-CORE STANDARD

Caution: **WARNING: The intra-building port(s) of the equipment or subassembly is suitable for connection to intrabuilding or unexposed wiring or cabling only. The intra-building port(s) of the equipment or subassembly MUST NOT be metallically connected to interfaces that connect to the OSP or its wiring. These interfaces are designed for use as intra-building interfaces only (Type 2 or Type 4 ports as described in GR-1089-CORE, Issue 4) and require isolation from the exposed OSP cabling. The addition of Primary Protectors is not sufficient protection in order to connect these interfaces metallically to OSP wiring.**

EC Declaration of Conformity

According to EN 45014:1998

Manufacturer's Name: Emerson Network Power
Embedded Computing

Manufacturer's Address: 8310 Excelsior Drive
Madison, Wisconsin 53717

Declares that the following product, in accordance with the requirements of 2004/108/EEC, EMC Directive and 1999/5/EC, RTTE Directive and their amending directives,

Product: ATCA Blade

Model Name/Number: ATCA-9305/10009986-xx

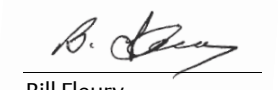
has been designed and manufactured to the following specifications:

EN55022:1998 Information Technology Equipment, Radio disturbance characteristics, Limits and methods of measurement

EN55024:1998 Information Technology Equipment, Immunity characteristics, Limits and methods of measurement

EN300386 V.1.3.2:2003-5 Electromagnetic compatibility and radio spectrum matters (ERM); Telecommunication network equipment; EMC requirements

As manufacturer we hereby declare that the product named above has been designed to comply with the relevant sections of the above referenced specifications. This product complies with the essential health and safety requirements of the EMC Directive and RTTE Directive. We have an internal production control system that ensures compliance between the manufactured products and the technical documentation.



Bill Fleury
Compliance Engineer

Issue date: April 7, 2009

CE

EMERSON
Network Power

Regulatory Agency Warnings & Notices (continued)

Regulatory Agency Warnings & Notices (continued)

Contents

Contents (continued)

Contents (continued)

Contents (continued)

Figures

(blank page)

Tables

Tables (continued)

Tables (continued)

(blank page)

Registers

(blank page)

Overview

The ATCA-9305 is an Advanced Telecom Computing Architecture (AdvancedTCA®, ATCA®) blade based on dual Cavium OCTEON™ CN5860 processors and the Freescale™ Semiconductor MPC8548 management processor. This blade is targeted at security and packet-processing applications in the wireless and transport market segments. These markets include data-plane packet-processor, security co-processor, video compression, and pattern matching.

The ATCA-9305 complies with the SCOPE recommended profile for central office ATCA systems, PICMG® 3.0 ATCA mechanical specifications, E-keying, and Hot Swap.

COMPONENTS AND FEATURES

The following is a brief summary of the ATCA-9305 hardware components and features:

Cavium Processor : The Cavium CN5860 processor is a highly programmable, high-performance 16-core architecture operating up to 800 MHz.

Management Processor: The Freescale PowerQUICC™ III MPC8548 processor is a 32-bit enhanced e500 core operating at 1 GHz.

Ethernet Switch: The Broadcom® BCM56802 is a sixteen-port, 10 GbE switch which interconnects the processors using SPI to XAUI™ bridges. The functionality includes both 10-Gbps XAUI and 1-Gbps SGMII PHY interfaces.

Stratix™ GX Bridge: There are two packet routing Altera® SPI-4.2 high-speed interconnect to XAUI bridges per CN5860 processor.

Ethernet: 10/100/1000BASE-T Ethernet ports are accessible via the front panel RJ45 connectors and through the base channel on the back panel. The 10 GbE ports route to the back panel through the fabric and RTM connectors.

Serial Port: The front panel serial port (MGT CSL) connects to the MPC8548 management processor.

System Management: This product supports an Intelligent Platform Management Controller (IPMC) based on a proprietary BMR-H8S-AMCc® reference design from Pigeon Point Systems. The IPMC has an inter-integrated circuit (I2C) controller to support an Intelligent Management Platform Bus (IPMB) that routes to the AdvancedTCA connector. The IPMB allows for features such as remote shutdown, remote reset, payload voltage monitoring, temperature monitoring, and access to Field Replaceable Unit (FRU) data.

PCI/PCIe: The PCI bus allows for read/write memory access between the MPC8548 processor, Ethernet switch, and Cavium processors. The four lane PCI Express® (PCIe) routes between the MPC8548 and the optional RTM.

Real-time Clock: The STMicroelectronics M41T00S RTC provides counters for seconds, minutes, hours, day, date, month, years, and century. The M41T00S serial interface supports I^2C bus and has a super-cap backup capable of maintaining the clock for a minimum of two hours.

Software: The Cavium CN5860 processor provides a GNU compiler that implements the MIPS64 Rel 2 instruction set in addition to the specialized instructions and a Linux® Board Specific Package (BSP) including the IP-stack optimization. The CN5860 also provides libraries that take advantage of the chip's hardware acceleration for certain security protocols.

RTM (optional): This blade supports a custom Rear Transition Module (RTM) with the following I/O:

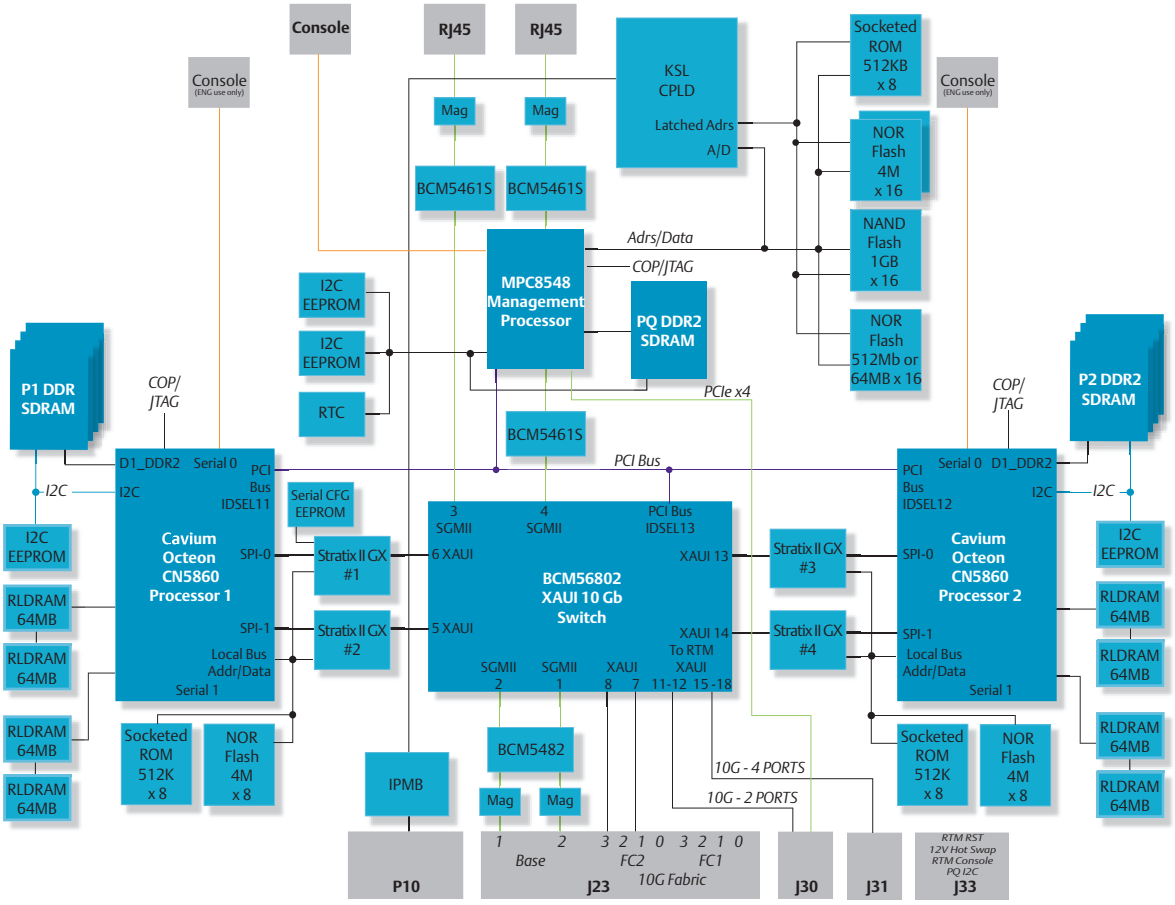
- Either two or six 10GbE connections
- One x4 PCI Express port from the MPC8548
- Connections for an MMC to control Hot Swap
- MPC8548 console port

For more detailed information, see the *ATCA-9305 Rear Transition Module User's Manual.*

FUNCTIONAL OVERVIEW

The following block diagram provides a functional overview for the ATCA-9305:

Figure 1-1: *General System Block Diagram*



ADDITIONAL INFORMATION

This section lists the ATCA-9305 hardware's regulatory certifications and briefly discusses the terminology and notation conventions used in this manual. It also lists general technical references.

Mean time between failures (MTBF) has been calculated at 439,924 hours using the Telcordia SR-332, Issue 1 (Reliability Prediction for Electronic Equipment), method 2 at 30° C.

Product Certification

The ATCA-9305 hardware has been tested to comply with various safety, immunity, and emissions requirements as specified by the Federal Communications Commission (FCC), Underwriters Laboratories (UL), and others. The following table summarizes this compliance:

Table 1-1: *Regulatory Agency Compliance*

Type:	Specification:
Safety	IEC60950/EN60950 — Safety of Information Technology Equipment (Western Europe) UL60950, CSA C22.2 No. 60950 — Safety of Information Technology Equipment, including Electrical Business Equipment (BI-National) GR1089-CORE Global IEC — CB Scheme Report IEC 60950, all country deviations
Environmental	NEBS: Telecordia GR-63 — Section 4.1.1 Transportation and Storage Environmental Criteria; Section 4.1.2 Operating Temperature and Humidity; Section 4.1.3 Altitude; Section 4.1 4 Temperature Margins; Section 4.4.1 Earthquake Environment; Section 4.4.4 Office Vibration: Section 4.4.5 Transportation Vibration

Type:	Specification: (continued)
EMC	FCC Part 15, Class A— Title 47, Code of Federal Regulations, Radio Frequency Devices
	ICES 003, Class A — Radiated and Conducted Emissions, Canada
	NEBS: Telecordia GR-1089 level 3 — Emissions and Immunity (circuit pack level testing only)
	EN55022 — Information Technology Equipment, Radio Disturbance Characteristics, Limits and Methods of Measurement
	EN55024 — Information Technology Equipment, Immunity Characteristics, Limits and Methods of Measurement
	ETSI EN300386 — Electromagnetic Compatibility and Radio Spectrum Matters (ERM), Telecommunication Network Equipment, Electromagnetic Compatibility (EMC) Requirements
	AS/NZS 3548 003, Class A — Standard for radiated and conducted emissions for Australia and New Zealand

Emerson maintains test reports that provide specific information regarding the methods and equipment used in compliance testing. Unshielded external I/O cables, loose screws, or a poorly grounded chassis may adversely affect the ATCA-9305 hardware's ability to comply with any of the stated specifications.

The UL web site at ul.com has a list of Emerson's UL certifications. To find the list, search in the online certifications directory using Emerson's UL file number, E190079. There is a list for products distributed in the United States, as well as a list for products shipped to Canada. To find the ATCA-9305, search in the list for 10009986-*xx*, where *xx* changes with each revision of the printed circuit board.

The Ethernet connection of the equipment or subassembly must be connected with shielded cables that are grounded at both ends.

RoHS Compliance

The ATCA-9305 is compliant with the European Union's RoHS (Restriction of use of Hazardous Substances) directive created to limit harm to the environment and human health by restricting the use of harmful substances in electrical and electronic equipment. Effective July 1, 2006, RoHS restricts the use of six substances: cadmium (Cd), mercury (Hg), hexavalent chromium (Cr (VI)), polybrominated biphenyls (PBBs), polybrominated diphenyl ethers (PBDEs) and lead (Pb). Configurations that are RoHS compliant are built with lead-free solder.

To obtain a certificate of conformity (CoC) for the ATCA-9305, send an e-mail to sales@artesyncp.com or call 1-800-356-9602. Have the part number(s) (e.g., C000####-##) for your configuration(s) available when contacting Emerson.

Terminology and Notation

Active low signals: An active low signal is indicated with an asterisk * after the signal name.

Byte, word: Throughout this manual *byte* refers to 8 bits, *word* refers to 16 bits, and *long word* refers to 32 bits, *double long word* refers to 64 bits.

PLD: This manual uses the acronym, *PLD*, as a generic term for programmable logic device (also known as FPGA, CPLD, EPLD, etc.).

Radix 2 and 16: Hexadecimal numbers end with a subscript *16*. Binary numbers are shown with a subscript *2*.

Technical References

Further information on basic operation and programming of the ATCA-9305 components can be found in documents listed in Table 1-2.

Table 1-2: *Technical References*

Device / Interface:	Document: [1]
ATCA	*AdvancedTCA® Base Specification* (PICMG® 3.0 Revision 2.0 March 18, 2005) *Engineering Change Notice 3.0-1.0-001* (PICMG® 3.0 R2.0: ECN 3.0-2.0-001 June 15, 2005) *Ethernet/Fibre Channel for AdvancedTCA™ Systems* (PICMG® 3.1 Revision 1.0 January 22, 2003) http://www.picmg.org
CPU CN5860 MPC8548	*Cavium Networks OCTEON™ Plus CN58XX Hardware Reference Manual* (Cavium Networks, CN58XX-HM-1.2 Sept. 2008) http:/www.caviumnetworks.com
	MPC8548E PowerQUICC™ III Integrated Processor Family Reference Manual (Freescale™ Semiconductor, Inc. MPC8548ERM Rev.2, 02/2007 http://www.freescale.com
DRAM	*576Mb: x9, x18, x36 2.5V V_{EXT}, 1.8C V_{DD}, HSTL, CIO,RLDRAM II Data Sheet* (Micron Technology, Inc. 576Mb_RLDRAM_II_CIO_D1.fm - Rev C 9/07 EN) http://www.micron.com
EEPROM	*Atmel® 2-Wire Serial EEPROM 64K (8192 x 8) Preliminary Data Sheet* (Atmel Corporation, 5174C-SEEPR-6/07) http://www.atmel.com
Ethernet BCM5461S BCM5482	*10/100/1000BASE-T Gigabit Ethernet Transceiver Data Sheet* (Broadcom® Corporation, Document 5461S-DS17-R 5/12/08) *10/100/1000BASE-T Gigabit Ethernet Transceiver Data Sheet* (Broadcom® Corporation, Document 5482-DS04-R 10/18/07) http://www.broadcom.com

Device / Interface:	Document: [1] (continued)
Flash	*32 Mbit (x8/x16) Concurrent SuperFlash Data Sheet* (Silicon Storage Technology, Inc., S71270-01-000 9/05) http://www.sst.com *mDOC H3 Embedded Flash Drive (EFD) featuring Embedded TrueFFS® Flash Management Software Preliminary Data Sheet* (msystems 92-DS-1205-10 Rev. 0.2 June 2006) http://www.m-systems.com/mobile *StrataFlash® Embedded Memory (P33) Data Sheet* (Intel®, Order Number: 314749-004 November 2007) http://www.intel.com *4. Serial Configuration Devices (EPCS1, EPCA4, EPCS16, & EPCS64)* (Altera® Corporation CS1014-2.0 April 2007) http://www.altera.com
IPMI	*IPMI – Intelligent Platform Management Interface Specification v2.0* (Intel Hewlett-Packard NEC Dell, Rev. 1.0, Feb. 12, 2004) *IPMI – Intelligent Platform Management Bus Communications Protocol Specification v1.0* (Intel Hewlett-Packard NEC Dell, Rev. 1.0, November 15, 1999) *IPMI – Platform Management FRU Storage Definition v1.0* (Intel Hewlett-Packard NEC Dell, Rev. 1.1, September 27, 1999) http://www.intel.com/design/servers/ipmi/ *Hardware Platform Management IPM Controller Firmware Upgrade Specification v1.0* (PICMG HPM.1 R1.0 May 4, 2007) http://www.picmg.org
RTC M41T00S	*Serial Access Real-Time Clock Data Sheet* (STMicroelectronics December 2004)
Switch BCM56802	*BCM56800 Series 20-Port 10-Gigabit Ethernet Multilayer Switch Preliminary Data Sheet* (Broadcom® Corporation, Document 56800-DS03-R 12/28/07) http://www.broadcom.com

1. Frequently, the most current information regarding addenda/errata for specific documents may be found on the corresponding web site.

(blank page)

Setup

This chapter describes the physical layout of the boards, the setup process, and how to check for proper operation once the boards have been installed. This chapter also includes troubleshooting, service, and warranty information.

ELECTROSTATIC DISCHARGE

Before you begin the setup process, please remember that electrostatic discharge (ESD) can easily damage the components on the ATCA-9305 hardware. Electronic devices, especially those with programmable parts, are susceptible to ESD, which can result in operational failure. Unless you ground yourself properly, static charges can accumulate in your body and cause ESD damage when you touch the board.

Caution: **Use proper static protection and handle ATCA-9305 boards only when absolutely necessary. Always wear a wriststrap to ground your body before touching a board. Keep your body grounded while handling the board. Hold the board by its edges—do not touch any components or circuits. When the board is not in an enclosure, store it in a static-shielding bag.**

To ground yourself, wear a grounding wriststrap. Simply placing the board on top of a static-shielding bag does not provide any protection—place it on a grounded dissipative mat. Do not place the board on metal or other conductive surfaces.

ATCA-9305 CIRCUIT BOARD

The ATCA-9305 circuit board is an ATCA blade assembly and complies with the PICMG 3.0 ATCA mechanical specification. It uses a 16-layer printed circuit board with the following dimensions:

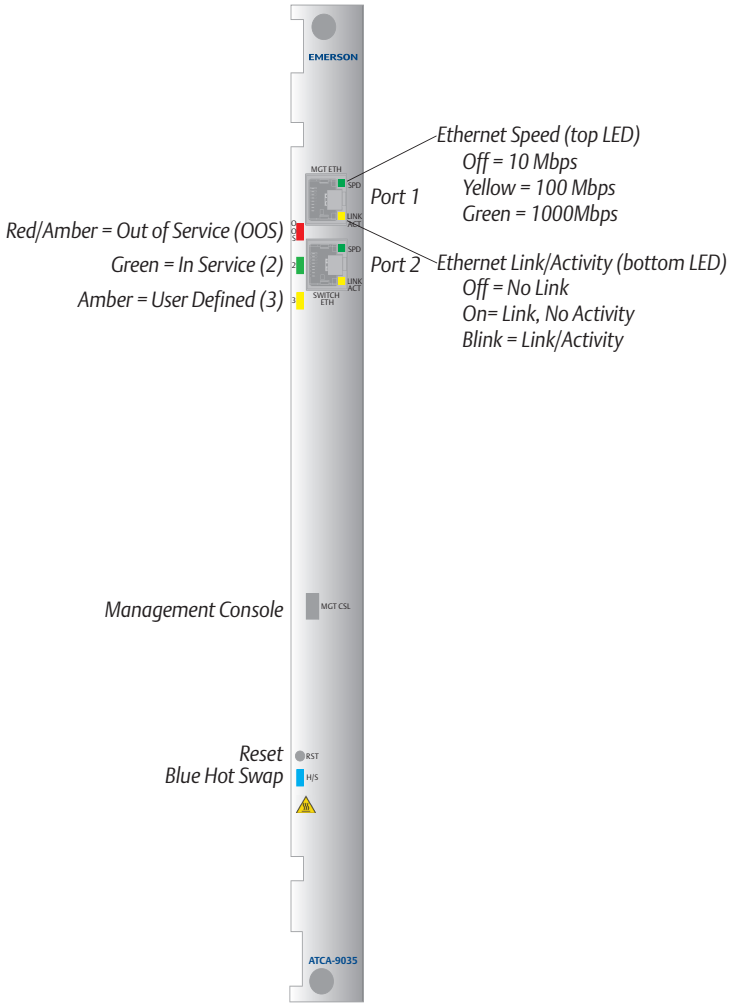
Table 2-1: *Circuit Board Dimensions*

Width:	Depth:	Height:	Weight (typical):
12.687 in. (322.25 mm)	11.024 in. (280.01 mm)	< .84 in. (<21.33 mm)	4.2 lb. (1.91 kg)[1]

1. This is the typical weight for the ATCA-9305. Board weight varies slightly per configuration; contact Technical Support if you require a specific configuration weight.

The following figures show the front panel, component maps, and LED locations for the ATCA-9305 circuit board.

Figure 2-1: *ATCA-9305 Front Panel*



Note: *The electromagnetic compatibility (EMC) tests used an ATCA-9305 model that includes a front panel assembly from Emerson Network Power, Embedded Computing.*

Caution: **For applications where the ATCA-9305 is provided without a front panel, or where the front panel has been removed, your system chassis/enclosure must provide the required electromagnetic interference (EMI) shielding to maintain CE compliance.**

Figure 2-2: *Component Map, Top (Rev. 01)*

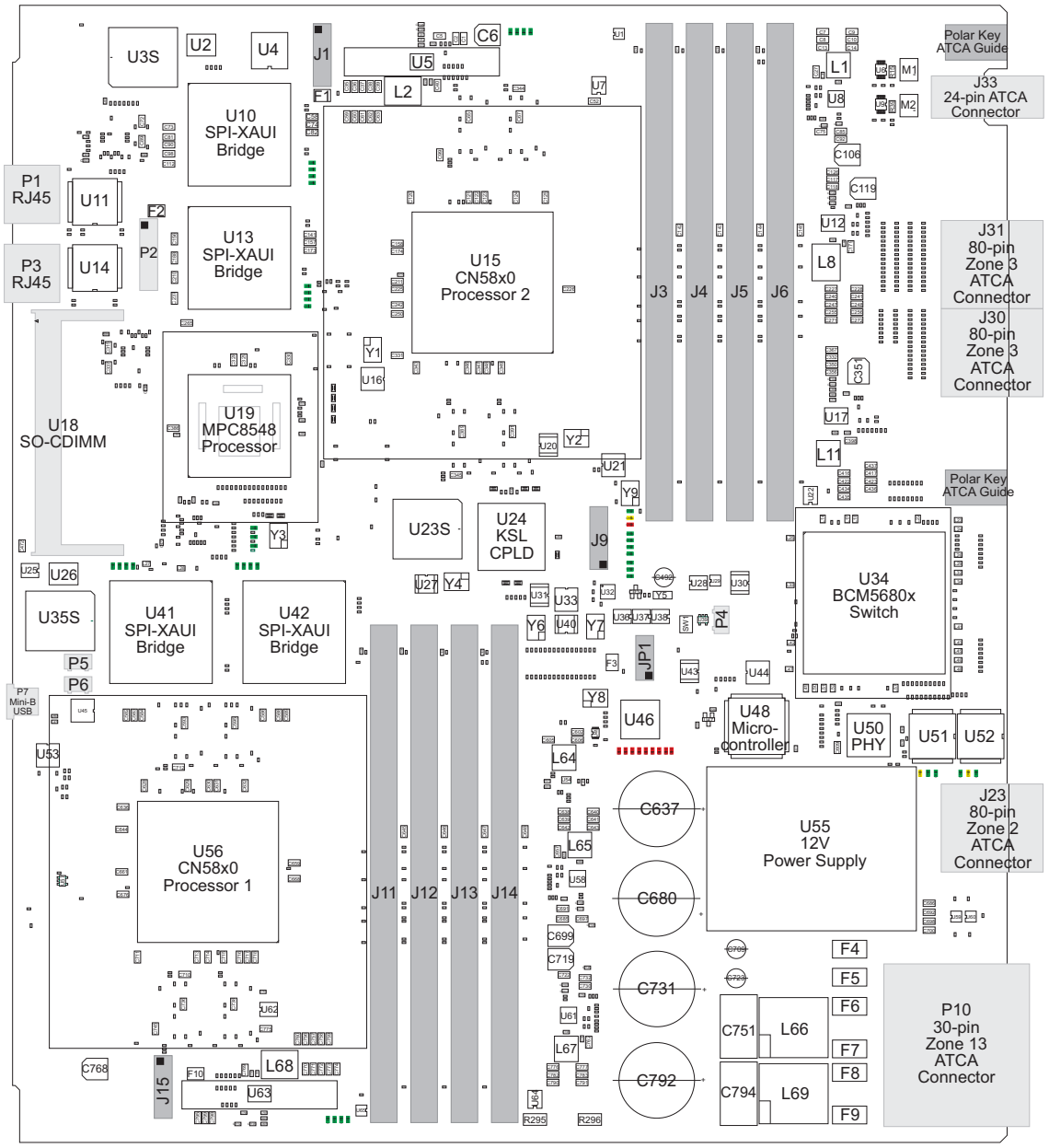


Figure 2-3: *Component Map, Bottom (Rev. 01)*

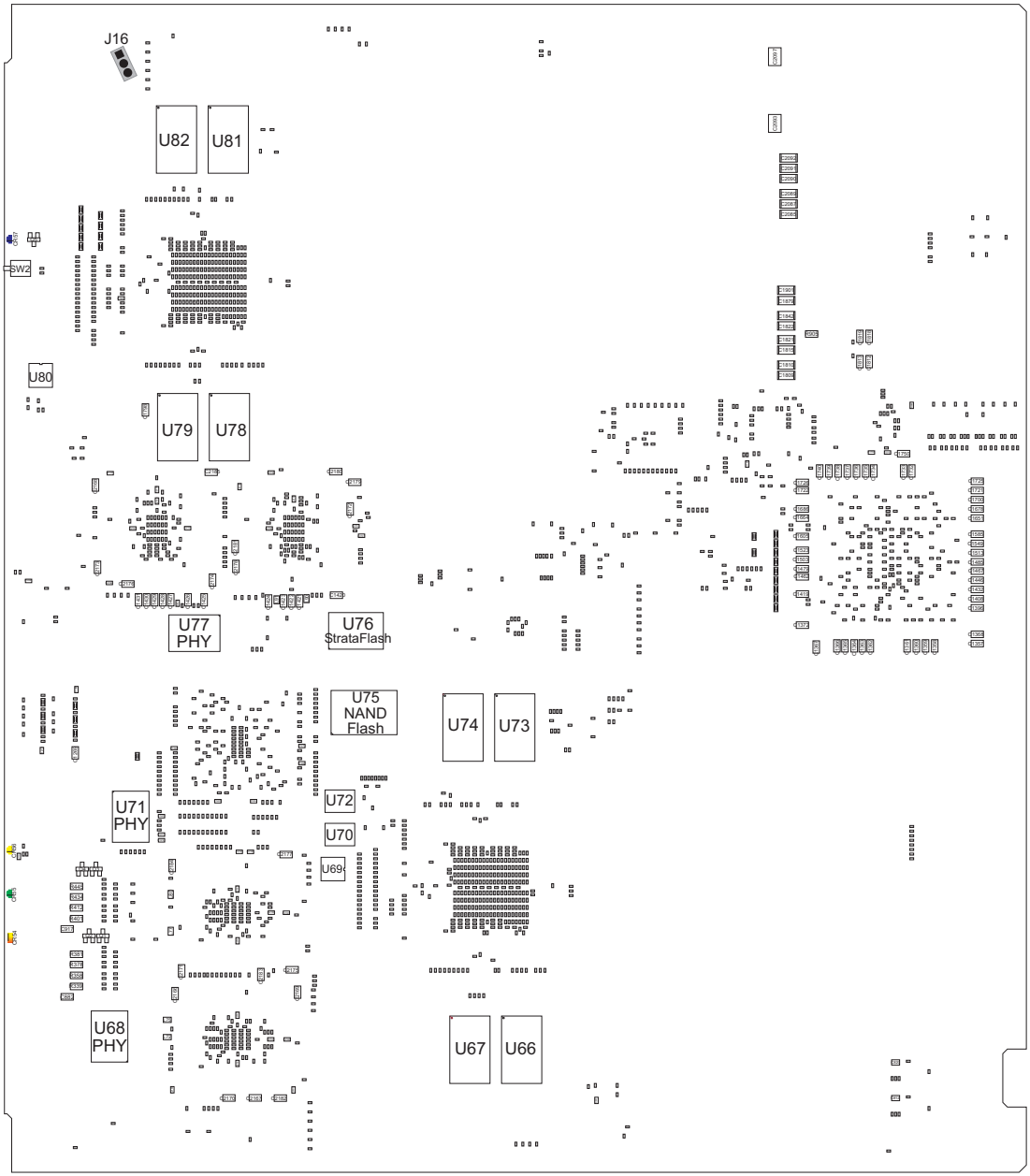


Figure 2-4: *LED, Fuse and Switch Locations, Top*

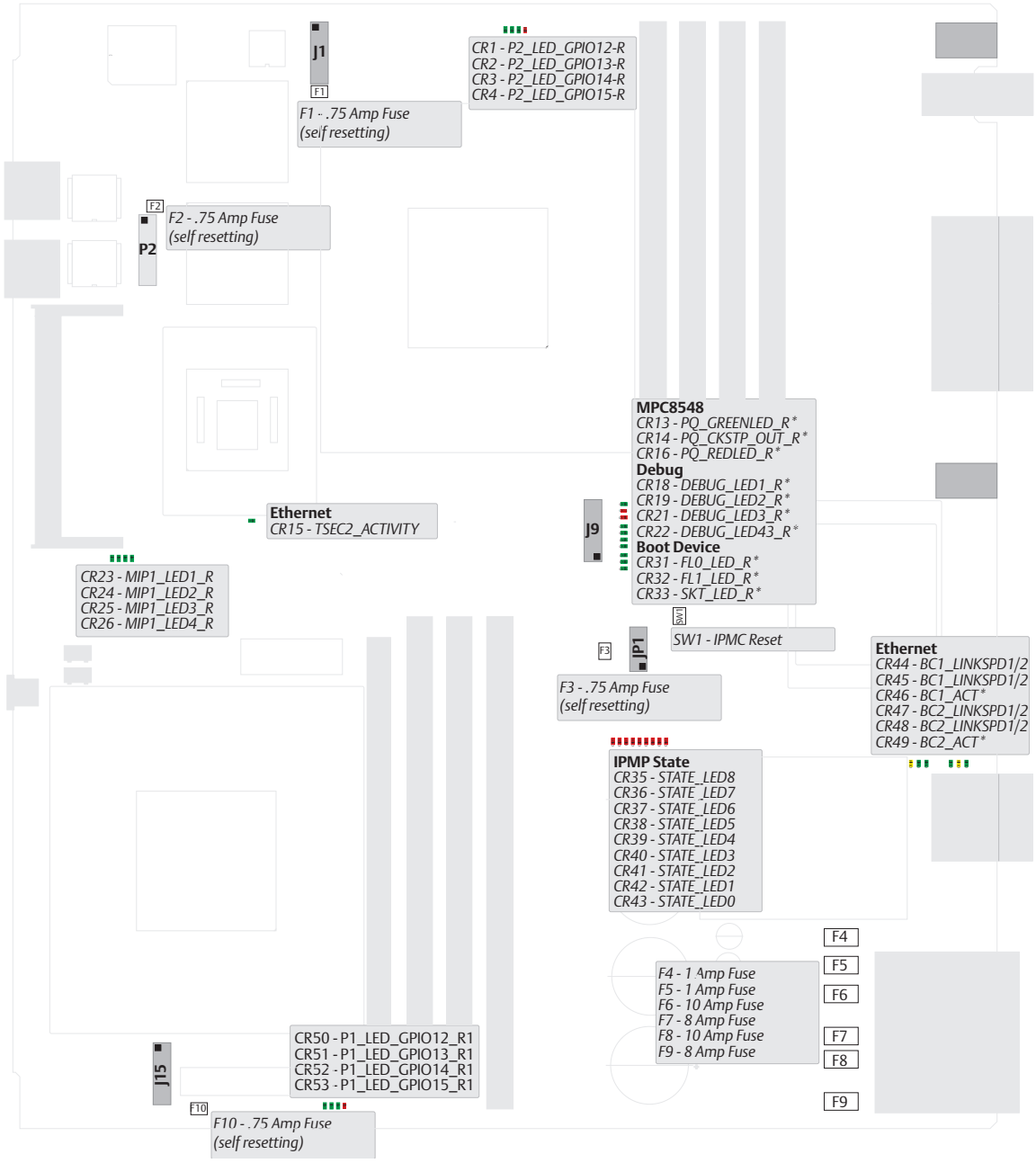
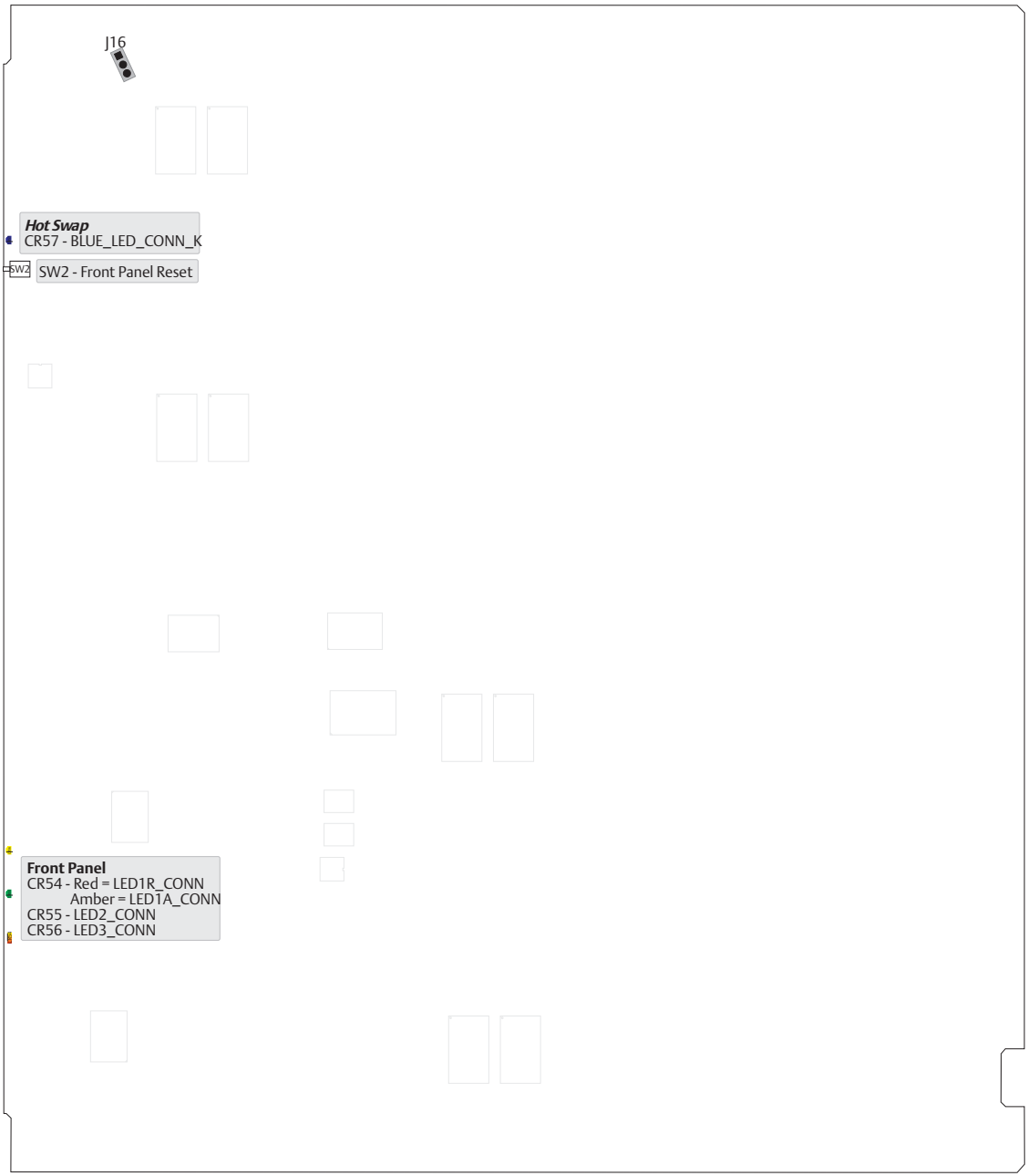


Figure 2-5: *LED and Switch Locations, Bottom*



Connectors

The ATCA-9305 circuit board has various connectors and headers (see the figures beginning on page 2-3), summarized as follows:

J1: This 14-pin JTAG header is used for debugging CN5860 processor 2. See Table 3-7.

J3-J6: These 240-pin sockets are installed for the CN5860 processor 1 DDR2 SDRAM memory.

J9: This 14-pin configuration header allows selection of boot device, and MPC8548 configuration for the configuration SROM. See Fig. 2-6.

J11-J14: These 240-pin sockets are installed for the CN5860 processor 2 DDR2 SDRAM memory.

J15: This 14-pin JTAG header is used for debugging CN5860 processor 1. See Table 3-7.

J23: The 80-pin Zone 2 connector provides 1 GB and 10 GB Ethernet access to the backplane, see Table 8-2.

J30-J31: The 80-pin Zone 3 connectors route PCIe and XAUI (10G) to the optional RTM. See Table 8-3 and Table 8-4 for pin assignments.

J33: The 24-pin Zone 3 connector routes the reset, Hot Swap, MPC8548 console, power, and IPMC I^2C to the optional RTM, see Table 8-5.

JP1: This is the 10-pin programming header for the IPMP, CPLD, and SPI 10G (1-4) devices, see Table 7-51.

P1: This 14-pin RJ45 connector with LEDs routes the Three-speed Ethernet Controller (TSEC1) between the MPC8548 and the front panel. See Table 6-4 for pin assignments.

P2: This 16-pin JTAG debug header accesses the MPC8548 processor, see Table 4-7.

P3: This 14-pin RJ45 connector with LEDs routes Ethernet (FP1) between the switch and the front panel, see Table 6-4 for pin assignments.

P4: The 5-pin vertical mini-B USB provides the IPMP EIA-232 console debug, see Table 7-52.

P5, P6: These 5-pin vertical mini-B USBs are the CN5860 console and for factory debug use only.

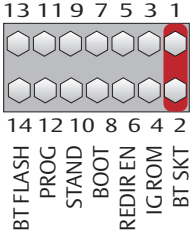
P7: This 5-pin mini-B USB is the console serial port for the MPC8548 management processor, see Table 4-8.

P10: The 30-pin Zone 1 connector routes IPMB to the backplane, see Table 8-1.

Configuration Header

There are a total of seven jumper pairs on J9 (pins 11-14 are spare posts). See figure Fig. 2-2 for the jumper location on the ATCA-9305. Also reference the "Jumper Settings (0x18)" register.

Figure 2-6: *Configuration Header, J9*



BT SKT: A shunt on pins 1-2 selects the 512 KB socketed ROM as the boot device for the MPC8548.

IG SROM: If the serial ROM configuration jumper is installed (pins 3-4), the ATCA-9305 will not try to configure (IGNORE_SROM*) from the MPC8548 serial ROM.

REDIR EN: A shunt installed on pins 5-6 disables the boot redirection, see page 7-41 for more information.

BOOT: A shunt on pins 7-8 causes both Cavium CN5860s to boot from their local bus and not boot over PCI.

STAND: A shunt on pins 9-10, IPMC stand alone mode, allows the board to boot without management control.

PROG: Installing a shunt on pins 11-12 puts the IPMC controller into programming mode. This is only used in the factory to configure the IPMC.

BT FLASH: If BOOT shunt is installed (booting from local bus), this shunt determines whether the boot is from local flash or socket. When this BT FLASH shunt is installed, the ATCA-9305 boots from flash. Otherwise, it boots from the socket.

ATCA-9305 SETUP

You need the following items to set up and check the operation of the Emerson ATCA-9305:

ATCA chassis and power supply

MPC8548 Console cable for EIA-232 port, Emerson part # C0007662-00

Computer terminal

Save the antistatic bag and box for future shipping or storage.

Power Requirements

The ATCA-9305 circuit board uses –48 volts from the backplane to derive 3.3 volts for the IPMC and 12 volts for payload power.

Table 2-2: *Typical Power Requirements*

Configuration:	Power:
1.0 GHz MPC8548 and 800 MHz Cavium processors, board running at room temperature with all processors at U-Boot prompt	135 watts

The exact power requirements for the ATCA-9305 circuit board depend upon the specific configuration of the board, including the CPU frequency and amount of memory installed on the board. Please contact Emerson Technical Support at 1-800-327-1251 if you have specific questions regarding the board's power requirements.

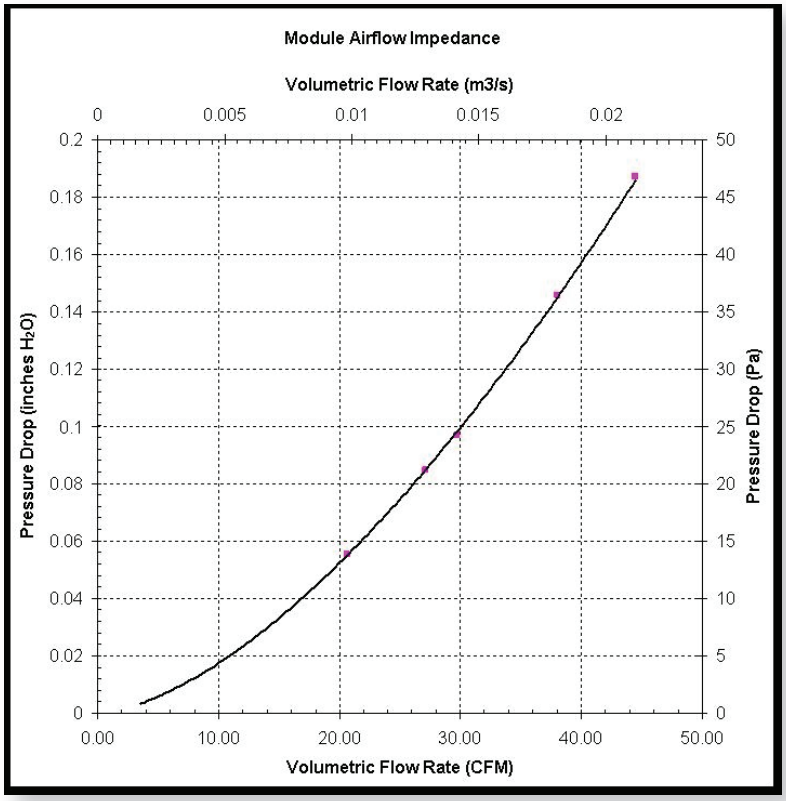
Environmental Considerations

As with any printed circuit board, be sure that air flow to the board is adequate. Chassis constraints and other factors greatly affect the air flow rate. The environmental requirements are as follows:

Table 2-3: *Environmental Requirements*

Environment:	Range:	Relative Humidity:
Operating Temperature	0° to +55° Centigrade, ambient (at board)	Not to exceed 85% (non-condensing)
Storage Temperature	–40° to 85° Centigrade	Not to exceed 95% (non-condensing)
Altitude	0 to 4,000 meters above sea level	–
Air Flow	Requires 30 CFM at 55° Centigrade at sea level. Meets thermal performance requirements of CP-TA ATCA ICD Book 1.1Class B-2	

Figure 2-7: *Air Flow Graph*



Hot Swap

The ATCA-9305 can be Hot Swapped, as defined in the AdvancedTCA specification (see reference in Table 1-2). This section describes how to insert and extract an ATCA-9305 module in a typical AdvancedTCA system. (These procedures assume the system is using a shelf manager.)

Note: *The ATCA-9305 Rear Transistion Module (RTM) has its own Hot Swap LED and switch, and it can be Hot Swapped in/out independently of the front board. If the front board is not present, then the RTM will not be powered. If the front board is Hot Swapped out, the RTM's blue LED will illuminate. In either case, the RTM can be safely removed.*

Insert a board:

1 Insert the ATCA-9305 into an available slot.

2 Push in the front panel handle (tab).

The blue Hot Swap LED on the front panel (see Fig. 2-1) flashes a long blink to indicate that board insertion is in progress and system management software is activating the slot. Then the blue LED turns off, indicating the insertion process is complete, and payload power is present.

Remove a board:

1 Pull out the handle (tab) on the ATCA-9305 front panel one click.

A short blink indicates the board is requesting permission for extraction.

2 Remove the board when the blue LED on the front panel is on (no payload power).

Caution: Do not remove the ATCA-9305 while the blue LED is blinking.

TROUBLESHOOTING

In case of difficulty, use the following checklist:

Be sure the ATCA-9305 circuit board is seated firmly in the carrier.

Be sure the system is not overheating.

Check the cables and connectors to be certain they are secure.

Check that your terminal is connected to a console port.

Technical Support

If you need help resolving a problem with your ATCA-9305, visit http://www.emersonembeddedcomputing.com/ on the internet or send E-mail to support@artesyncp.com. Please have the following information handy:

- ATCA-9305 serial number and product identification (see Fig. 2-8)
- MPC8548 monitor version number (see Fig. 9-1)
- Cavium monitor version number (see Fig. 3-3)
- version and part number of the operating system (if applicable)

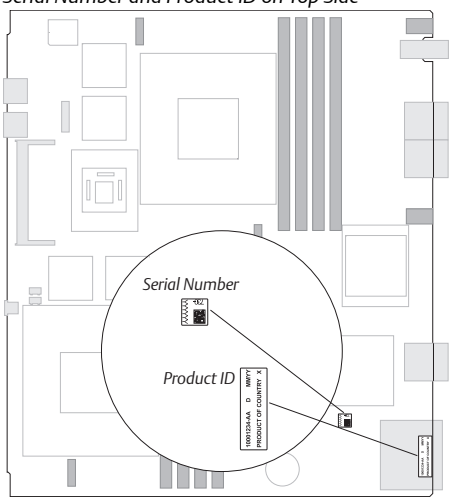
- whether your board has been customized for options such as a higher processor speed or additional memory
- license agreements (if applicable)

If you do not have internet access, please call Emerson for further assistance:

(800) 327-1251 or (608) 826-8006 (US)

44-131-475-7070 (UK)

Figure 2-8: *Serial Number and Product ID on Top Side*



Product Repair

If you plan to return the board to Emerson Network Power for service, visit http://www.emersonembeddedcomputing.com/ on the internet or send E-mail to serviceinfo@artesyncp.com to obtain a Return Merchandise Authorization (RMA) number. We will ask you to list which items you are returning and the board serial number, plus your purchase order number and billing information if your ATCA-9305 hardware is out of warranty. Contact our Test and Repair Services Department for any warranty questions. If you return the board, be sure to enclose it in an antistatic bag, such as the one in which it was originally shipped. Send it prepaid to:

Emerson Network Power, Embedded Computing
Test and Repair Services Department
8310 Excelsior Drive
Madison, WI 53717

RMA #_____

Please put the RMA number on the outside of the package so we can handle your problem efficiently. Our service department cannot accept material received without an RMA number.

Comments and Suggestions

We welcome and appreciate your comments on our documentation. We want to know what you think about our manuals and how we can make them better.

Mail comments to us by filling out the following online form:
http://www.emersonnetworkpowerembeddedcomputing.com/ **Contact Us > Online Form**

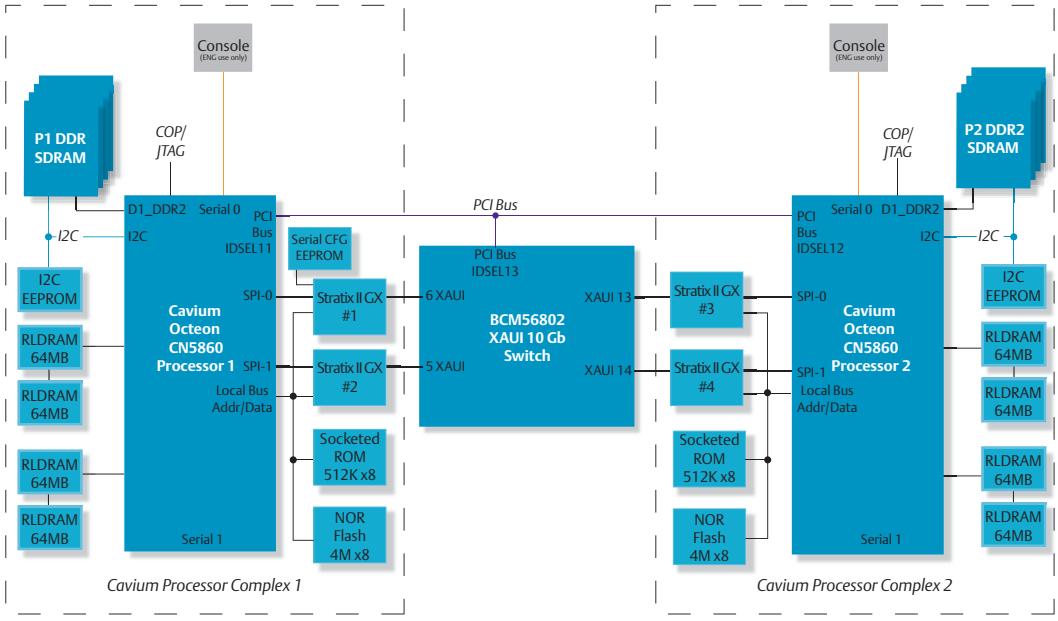
In "Area of Interest" select "Technical Documentation". Be sure to include the title, part number, and revision of the manual and tell us how you used it.

(blank page)

Cavium Processor Complex

The ATCA-9305 provides two Cavium processor complexes. The major devices on each complex consist of the Cavium CN5860 processor, two StratixGX bridges, SDRAM, RLDRAM®, an I^2C EEPROM, socketed ROM, Flash, and the PCI bus interface.

Figure 3-1: *Cavium Processor Complex Block Diagram*



CAVIUM CN5860 PROCESSOR

The main features of the CN5860 include:

Table 3-1: *CN5860 Features*

Feature:	Description:
Processor Core	Up to 16 cnMIPS™ cores
Core Speed Network Services Processor (NSP)	up to 800 MHz, processing up to 30 million packets per second
System Packet Interface	Two SPI-4.2 ports
L2 Cache	2 MB, eight-way set associative
DRAM	144-bit DDR2 DRAM interface
RLDRAM	18-bit RLDRAM, low-latency memory direct access
PCI	64-bit, PCI 2.3 compatible

The CN5860 and switch route packets using SPI-4.2 and control information flow using PCI. The CN5860 has two SPI-4.2 interfaces with each one supporting up to 16 ports. Two high-speed SPI-4.2 Altera (Stratix™ GX) FPGAs function as the SPI-to-XAUI bridge for each processor to switch complex. The PCI interface supports up to four ports, consequently a total of 36 ports can be supported internally by each CN5860.

Cavium Memory Map

Although the Cavium processors are 64-bit, the ATCA-9305 uses a 49-bit implementation. Refer to the *Cavium Networks OCTEON Plus CN58xx Hardware Reference Manual* for more detailed information on the memory map.

Table 3-2: *Cavium Address Summary*

Hex Physical Address:	Register Description:
1,2000,0000,0000	reserved
1,1F00,0000,0000	Cavium Hardware registers
1,1E00,0000,0000	PCI Memory Space (6)
1,1D00,0000,0000	PCI Memory Space (5)
1,1C00,0000,0000	PCI Memory Space (4)
1,1B00,0000,0000	PCI Memory Space (3)
1,1A00,0000,0000	PCI I/O Space
1,1910,0000,0000	reserved
1,1900,0000,0000	PCI Special Space
1,0700,0000,0000	CN58xx Registers
1,0001,0000,0000	reserved
1,0000,0000,0000	Local Boot Bus
0,0004,1000,0000	DDR2 SDRAM, middle block (256-512 MB)
0,0004,0000,0000	reserved
0,0000,2000,0000	DDR2 SDRAM, upper block (512 MB-2 GB)[1]
0,0000,1000,0000	reserved
0,0000,0000,0000	DDR2 SDRAM, bottom block (256 MB)

1. This depends on how much memory is installed.

PCI

The Cavium is a slave device on the PCI bus. The Cavium U-boot monitor image is provided by the MPC8548 management processor via PCI. The MPC8548 monitors the Cavium boot status and has the ability to try alternate boot images if the current one fails.

The CN5860 processor is designed such that another PCI device can initialize its memory interface, copy code over PCI into its local memory space, and then write a boot release register.

CN5860 Boot Over PCI

The PCI bus is configured to run at 66 MHz in 64-bit conventional PCI mode. On power-up, the CN5860 processor's 16 internal cores are held in reset. The MPC8548 management processor performs the following steps:

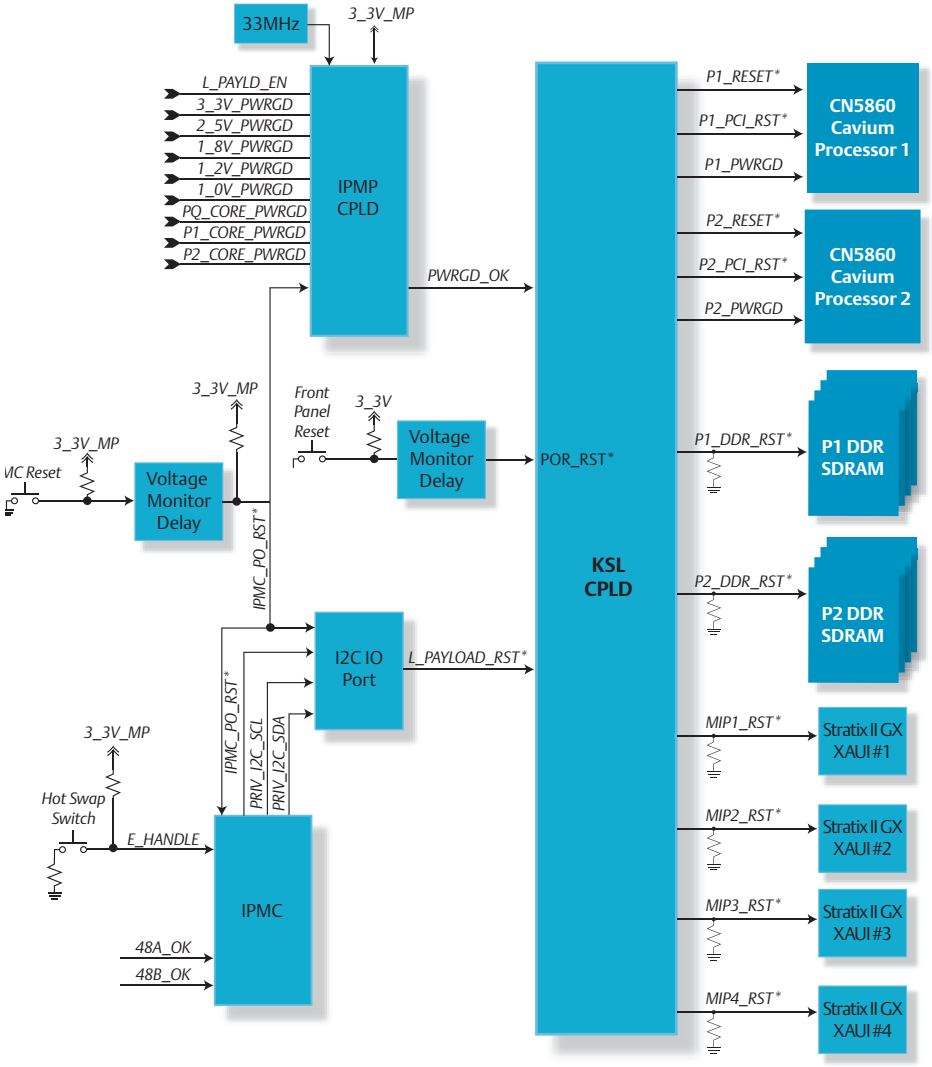
1. Initialize the CN5860 RAM.
2. Copy the CN5860 U-boot to the CN5860 RAM.
3. Copy boot code to the reset vector to jump to the U-boot code in RAM.
4. Release the CN5860 processor cores from reset.
5. Receive return codes from the CN5860 that indicate any boot or POST errors and take the appropriate action.

   The management processor (MPC8548) monitor implements a utility to load non-volatile memory redundant U-boot images for the CN5860 processors. The utility tags each copy as primary or secondary.

Cavium Reset

Each CN5860 can be reset independently of the other processor without affecting its operation. This task is performed by the MPC8548 management processor.

Figure 3-2: *CN5860 Reset Diagram*



CAVIUM ETHERNET

The Ethernet address for your board is a unique identifier on a network. The address consists of 48 bits (MAC [47:0]) divided into two equal parts. The upper 24 bits define a unique identifier that has been assigned to Emerson Network Power, Embedded Computing by IEEE. The lower 24 bits are defined by Emerson for identification of each of our products.

The Ethernet address for the ATCA-9305 is a binary number referenced as 12 hexadecimal digits separated into pairs, with each pair representing eight bits. The address assigned to the ATCA-9305 has the following form:

00 80 F9 xx yy zz

00 80 F9 is Emerson's identifier. The last three bytes of the Ethernet address consist of the port (one byte); 0x99(SPI 1), 0x9A (SPI 2), 0x9B (SPI 3), or 0x9C (SPI 4), followed by the serial number (two byte hexadecimal). The ATCA-9305 Cavium has been assigned the Ethernet address range 00:80:F9:99:00:00 to 00:80:F9:9C:FF:FF. The format is shown in Table 3-3.

Table 3-3: *Ethernet Port Address*

Offset:	MAC:	Description:	Ethernet Identifier (hex):
Byte 5	15:0	LSB of (serial number in hex)	–
Byte 4		MSB of (serial number in hex)	–
Byte 3	23:16	SPI 1 SPI 2 SPI 3 SPI 4	0x99 0x9A 0x9B 0x9C
Byte 2	47:24	Assigned to Emerson by IEEE	0xF9
Byte 1			0x80
Byte 0			0x00

The last two bytes, MAC[15:0], are calculated from the serial number stored in the Cavium EEPROM. This corresponds to the following formula: *n* –1000, where *n* is the unique serial number assigned to each board. So if an ATCA-9305 serial number is 1032, the calculated value is 32 (20_{16}), and the default Ethernet port addresses are:

- Cavium 1 SPI 1 MAC address is: 0x00 0x80 0xF9 0x99 0x00 0x20
- Cavium 1 SPI 2 MAC address is: 0x00 0x80 0xF9 0x9A 0x00 0x20
- Cavium 2 SPI 1 MAC address is: 0x00 0x80 0xF9 0x9B 0x00 0x20
- Cavium 2 SPI 2 MAC address is: 0x00 0x80 0xF9 0x9C 0x00 0x20

CAVIUM MONITOR

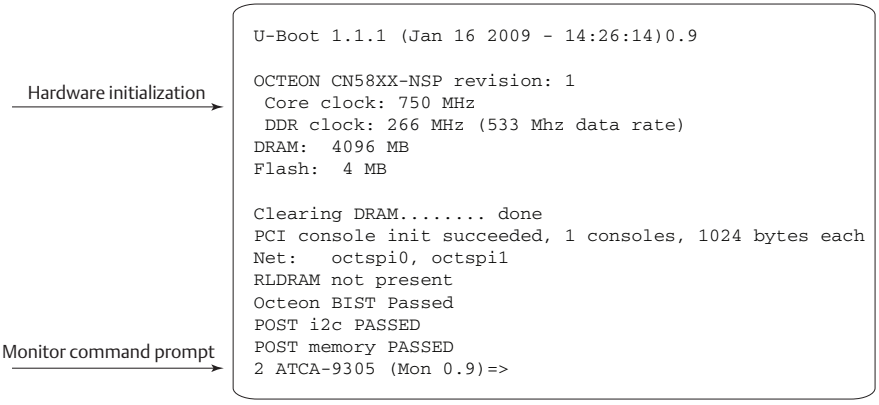
The primary function of the monitor software is to transfer control of the hardware to the user's application. Secondary responsibilities include:

- low-level initialization of the hardware
- diagnostic tests
- low-level monitor commands/functions to aid in debug

Start-up Display

At power-up or after a reset, the Cavium monitor runs diagnostics and reports the results in the start-up display, see an example in Fig. 3-3. During the power-up sequence, the monitor configures the board according to the environment variables (see "MPC8548 Environment Variables" on page 9-26). If the configuration indicates that autoboot is enabled, the monitor attempts to load the application from the specified device. If the monitor is not configured for autoboot or a failure occurs during power-up, the monitor enters normal command-line mode. The monitor command prompt in Fig. 3-3 is the result of a successful hardware boot of the ATCA-9305.

Figure 3-3: *Example Cavium CN5860 Monitor Start-up Display*

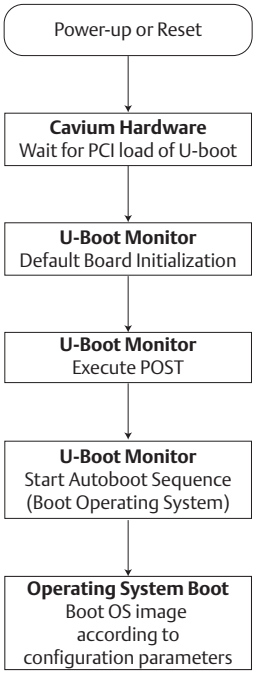


Note: *There will be either a 1 or 2 in front of the monitor prompt indicating which Cavium processor is prompting.*

Power-up/Reset Sequence

The Cavium CN5860 processor follows the boot sequence in Fig. 3-4 before auto-booting the operating system or application software. At power-up or board reset, the monitor performs hardware initialization, diagnostic routines, autoboot procedures, and if necessary, invokes the command line. See Table 3-5 for default Cavium environment variables settings.

Figure 3-4: *Power-up/Reset CN5860 Boot Sequence Flowchart*



Diagnostic Tests During Power-up and Reset

The Cavium monitor diagnostic tests can be executed during power-up or invoked from the monitor's command prompt. This is accomplished by changing the state of the monitor configuration parameters that define power-up and reset diagnostics mode. If the *poweron-diags* parameter is set to "on", the monitor invokes the diagnostic tests after a reset of the hardware. Results are displayed to the console including whether the test passed or failed.

POST Diagnostic Results

The ATCA-9305 Power-On Self-Test (POST) diagnostic results are stored as a 32-bit value in memory accessible by the management console at location 0x80080A6C. Each bit indicates the result of a specific test, so this field can store the results of up to 32 diagnostic tests. Table 3-4 assigns the bits to specific tests.

Table 3-4: *POST Diagnostic Results—Bit Assignments*

Bit:	Diagnostic Test:	Description:	Value:
0-1	Reserved		
2	DRAM	Verify address and data lines are intact	
3	Cavium BIST	-	0 Passed the test
4	I^2C	Verify all local I^2C devices are connected to the I^2C bus	1 Failure detected
5-31	Reserved		

Cavium Environment Variables

The following table lists the standard Cavium environment variables:

Table 3-5: *Standard Cavium Environment Variables*

Variable:	Default Value:	Description:
baudrate	115200	Console port baud rate Valid rates: 9600, 14400, 19200, 38400, 57600, 115200
bootcmd	" "	Command to execute when auto-booting or executing the 'bootd' command
bootdelay	0	Choose the number of seconds the Monitor counts down before booting user application code Valid options: time in seconds, -1 to disable autoboot
bootfile	" "	Path to boot file on server (used with TFTP)–set this to "path/file.bin" to specify filename and location of the file to load.
ethaddr	undefined	SPI 1 MAC address
eth1addr	undefined	SPI 2 MAC address
ethact	octspi0	Specifies Ethernet port to use
gatewayip	0.0.0.0	Select the network gateway machine IP address
hostname	none	Target hostname
ipaddr	0.0.0.0	Board IP address
loadaddr	0x20000000	Define the address to download user application code (used with TFTP)
netmask	0.0.0.0	Board sub-network mask
powerondiags	off	Turns POST diagnostics on or off after power-on/reset Valid options: on, off
rootpath	eng/	Path name of the NFS' server root file system
serial#	xxxxx	Board serial number
serverip	0.0.0.0	Boot server IP address
stderr	serial	Sets the standard destination for console error reporting Valid options: serial, pci

Variable:	Default Value:	Description: (continued)
stdin	serial	Sets the standard source for console input Valid options: serial, pci
stdout	serial	Sets the standard destination for console output Valid options: serial, pci

MEMORY

The processor complex supports DDR2 Synchronous DRAM (SDRAM) and Reduced Latency DRAM (RLDRAM) memory devices.

DDR2 SDRAM

The ATCA-9305 supports up to 16 gigabytes of 144-bit wide DDR2 SDRAM per processor complex. The SDRAM interface clock speed frequency is 400 MHz. The four low-profile, dual-inline memory modules (buffered DIMM) are installed in 240-pin very low profile (VLP) sockets to reduce board density and routing constraints. A 2 KB EEPROM on the DIMM provides the serial presence detection (SPD). On-card SDRAM occupies physical addresses from $0,0000,0000,0000_{16}$ to $0,0003,FFFF,FFFF_{16}$.

Each processor memory bus is operating in 144-bit mode. Error-correcting Code (ECC) is performed on the memory bus so that the CN5860 detects all double-bit errors, multi-bit errors within a nibble, and corrects all single-bit errors.

RLDRAM

Each CN5860 supports 256 MB Common I/O (CIO) RLDRAM operating up to 400 MHz (depends on the processor speed). The Micron RLDRAM II is organized as 32Mx18x8 internal banks. The DDR I/O interface transfers two data words per clock cycle. Output data is referenced to the free-running output data clock. Read and write accesses to the RLDRAM are burst-oriented. RLDRAM is accessed by using Cavium-specific instructions which operate on MIPS Coprocessor 2.

I^2C EEPROM

Each Cavium processor complex has one user EEPROM device for parameter storage located on the I^2C bus, address 0xA8. The I^2C bus for each processor is completely independent from the other CN5860 processor and MPC8548 processor I^2C buses. The Atmel two-wire serial EEPROM on each CN5860 processor I^2C interface consists of the Serial Clock (SCL) input and the Serial Data (SDA) bidirectional lines.

Table 3-6: *Cavium NVRAM Memory Map*

Address Offset (hex):	Description:	Window Size (bytes)
0x1E00-0x1FFF	Monitor parameters	256
0x0000-0x1D36	User defined	79F

Flash, 512 KB x 8

The 512 KB of 32-pin PLCC socketed flash starts at physical address1D46,0000$_{16}$ and is used for Engineering code. The StrataFlash features high-performance fast asynchronous access times, low power, and flexible security options.

Flash, 4 MB x 16

The 4 MB soldered NOR flash starts at physical address 1D05,0000$_{16.}$ The 32-Mbit device provides CN5860 code storage and non-volatile memory.

STRATIXGX INTERCONNECT

The Altera StratixGX FPGA provides the high-speed SPI-4.2 interconnect. Each complex has dual SPI-to-XAUI bridges connected to the XAUI Ethernet switch ports.

PLD Registers

The FPGA bridge is located at address 0x1D030000. Use the following registers to access the XAUI to SPI bridge configuration registers. See the “Read Example” and “Write Example.”

Data Registers

Register 3-1: *Data 31:24 (0x0)*

Bits:	R/W:	Function:
7	R/W	Data 31
6	R/W	Data 30
5	R/W	Data 29
4	R/W	Data 28
3	R/W	Data 27
2	R/W	Data 26
1	R/W	Data 25
0	R/W	Data 24

Register 3-2: *Data 23:16 (0x1)*

Bits:	R/W:	Function:
7	R/W	Data 23
6	R/W	Data 22
5	R/W	Data 21
4	R/W	Data 20
3	R/W	Data 19
2	R/W	Data 18
1	R/W	Data 17
0	R/W	Data 16

Register 3-3: *Data 15:8 (0x2)*

Bits:	R/W:	Function:
7	R/W	Data 15
6	R/W	Data 14
5	R/W	Data 13
4	R/W	Data 12
3	R/W	Data 11
2	R/W	Data 10
1	R/W	Data 9
0	R/W	Data 8

Register 3-4: *Data 7:0 (0x3)*

Bits:	R/W:	Function:
7	R/W	Data 7
6	R/W	Data 6
5	R/W	Data 5
4	R/W	Data 4
3	R/W	Data 3
2	R/W	Data 2
1	R/W	Data 1
0	R/W	Data 0

Address Registers

Register 3-5: *Address 9:8 (0x4)*

Bits:	R/W:	Function:
7	–	Reserved
6	–	
5	–	
4	–	
3	–	
2	–	
1	R/W	Address 9
0	R/W	Address 8

Register 3-6: *Address 7:0 (0x5)*

Bits:	R/W:	Function:
7	R/W	Address 7
6	R/W	Address 6
5	R/W	Address 5
4	R/W	Address 4
3	R/W	Address 3
2	R/W	Address 2
1	R/W	Address 1
0	R/W	Address 0

Control Register

The write only Control register performs two functions:

- Writing a value of 0x01 causes the contents of the Data registers to be written to the FPGA bridge at the location specified by the Address registers.
- Writing a value of 0x02 causes the contents of the Data registers to be overwritten by the contents of the FPGA bridge at the location specified by the Address registers.

Note: *Writing any other value to the Control register will be ignored.*

Register 3-7: *Control (0x6)*

Bits:	R/W:	Function:
7	–	Reserved
6	–	
5	–	
4	–	
3	–	
2	–	

Bits:	R/W:	Function:
1	W	Read
0	W	Write

Version Register

This read-only register tracks the PLD versions. The version is hard coded in the PLD and changes with every released code change. Version starts at 01_{16}.

Register 3-8: *Version (0x7)*

Bits:	R/W:	Function:
7	R	0x01
6	R	
5	R	
4	R	
3	R	
2	R	
1	R	
0	R	

Scratch Register

All registers in this range act as the same register.

Register 3-9: *Scratch (0x8-0x3F)*

Bits:	R/W:	Function:
7	R/W	
6	R/W	
5	R/W	
4	R/W	
3	R/W	
2	R/W	
1	R/W	
0	R/W	

Read Example: To read the FPGA bridge SPI_COMMAND register at 0x204, use the following commands.

Set address bits 9:8.

```
=>write64b 1d030004 02
```

Set address bits 7:0.

```
=>write64b 1d030005 04
```

Perform a read.

```
=>write64b 1d030006 02
```

Display the results.

```
=>read64l 1d030000
```

Write Example: To write to the FPGA bridge MAC_CMD_CFG register at 0x00C, use the following commands.

Set data bits 31:24.

```
=>write64b 1d030000 a9
```

Set data bits 23:16.

```
=>write64b 1d030001 b8
```

Set data bits 15:8.

```
=>write64b 1d030002 c7
```

Set data bits 7:0.

```
=>write64b 1d030003 d6
```

Set address bits 9:8.

```
=>write64b 1d030004 00
```

Set address bits 7:0.

```
=>write64b 1d030005 0c
```

Perform a write.

```
=>write64b 1d030006 01
```

HEADERS AND CONNECTORS

COP/JTAG Headers

The CN5860 processor complex uses headers J1 and J15 for debug.

Table 3-7: *CN5860 Processor COP/JTAG Headers*

Pin:	J1 (processor 2):	J15 (processor 1):
1	P2_ETRST*	P1_ETRST*
2	ground	ground
3	P2_TDI	P1_TDI
4	ground	ground

Pin:	J1 (processor 2):	J15 (processor 1): (continued)
5	P2_ETDO	P1_ETDO
6	ground	ground
7	P2_TMS	P1_TMS
8	ground	ground
9	P2_TCK	P1_TCK
10	ground	ground
11	P2_EJTAG_RST	P1_EJTAG_RST
12	key (pin not installed)	key (pin not installed)
13	P2_EJTAG_DINT	P1_EJTAG_DINT
14	P2_COP_PWR (3.3V)	P1_COP_PWR (3.3V)

Console Serial Ports (optional)

Connectors P6 (processor P1) and P5 (processor P2) access the CN5860 processors for Engineering debug use only. The supported baud rates for these ports operate at 9600, 14400, 19200, 38400, 57600, and 115200 bps. (The default rate is 115200 bps.)

Table 3-8: *CN5860 Processor Debug Headers*

Pin:	P6:	P5:
1	no connect	no connect
2	P1_SER1_RXD	P2_SER1_RXD
3	P1_SER1_TXD	P2_SER1_TXD
4	no connect	no connect
5	signal ground	signal ground
6-7	shield	signal ground

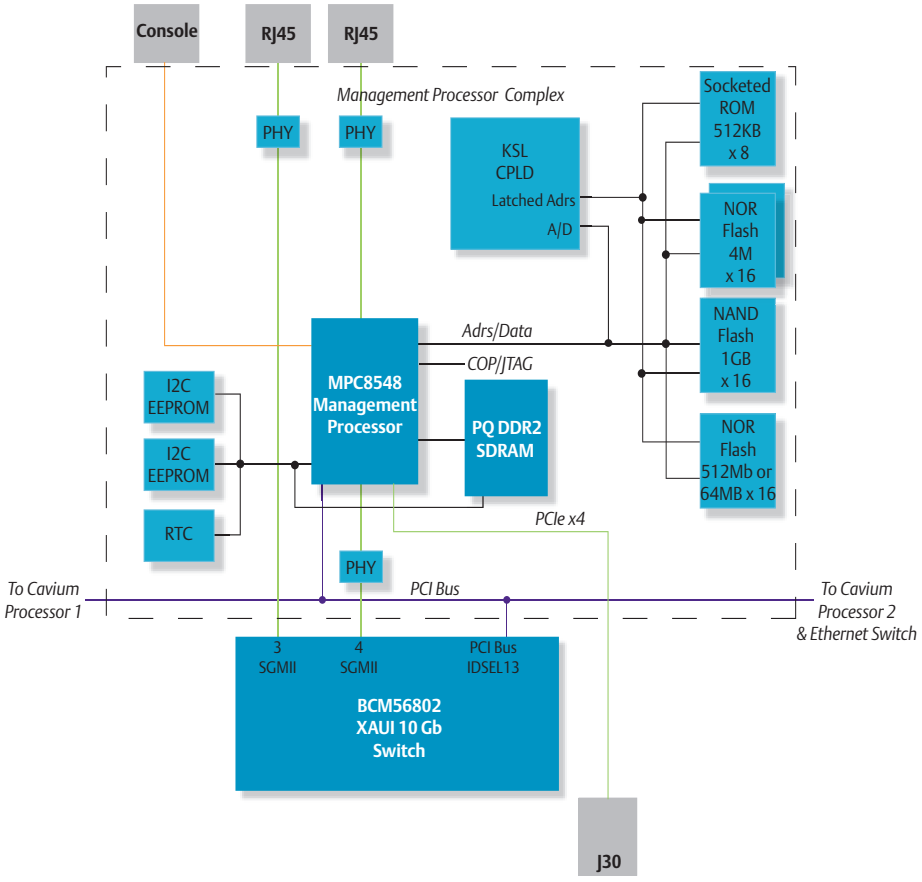
(blank page)

Management Complex

The ATCA-9305 management complex is comprised of the Freescale MPC8548 processor, CPLD, SDRAM, flash, I^2C EEPROM, Real-time Clock, and PCI bus interface. Board power-up, booting and monitoring the Cavium processors, PCI bus arbitration, interrupt servicing, memory persistence functionality, and other board level management tasks are implemented using the MPC8548 processor. The MPC8548 stores the Cavium operating system and monitor code in its local memory and then uses the boot over PCI functionality to bring up the Cavium processor complexes. The CPLD registers are described in Chapter 5. See Chapter 9 for the Management Processor Monitor.

The management complex connects to the Broadcom Ethernet switch via a 1000BASE-T Ethernet port. This connection uses the TSEC2 interface operating in SGMII mode. See Chapter 6, “Ethernet Interface.”

Figure 4-1: *MPC8548 Management Processor Complex Block Diagram*



MPC8548 PROCESSOR

The MPC8548 processor has the following features:

Table 4-1: *MPC8548 Features*

Feature:	Description:
L1 Cache	32-kilobyte data and instruction caches with parity protection, 32-byte line, eight-way set associative
L2 Cache	512 kilobytes, eight-way set associative
CPU Core Speed	1 GHz with a 400 MHz DDR2 bus
DDR2 Memory Controller	64-bit data interface, four banks of memory supported (each up to 4 GB), full ECC support
Dual I2C Controllers	Two-wire interface, master or slave I^2C support
Boot Sequencer	Loads configuration data from serial ROM at reset via the I^2C interface
Ethernet	Four 10/100/1000 enhanced three-speed controllers (eTSECs), full-/half-duplex support, MAC address recognition
Local Bus Controller (LBC)	DDR2 SDRAM memory controller, General Purpose Chip Select Machine (GPCM), three User-Programmable Machines (UPM), eight chip selects support eight external slaves
PCI	64-bit, PCI 2.2 compatible
PCI Express	Single x4 PCIe high-speed interconnect, complies with *PCI Express™ Base Specification Revision 1.0a*
JTAG	Complies with IEEE Std. 1149.1

For more detailed information, reference the Freescale *MPC8548E PowerQUICC™ III Integrated Processor Family Reference Manual.*

MPC8548 Memory Map

The monitor can boot from either the soldered flash (Bank 1, default) or the socketed PLCC device. Based on the configuration header (see page 2-8) either the socketed device or soldered flash is mapped to the boot bank at $FFF8,0000_{16}$, see Fig. 4-2. Information on particular portions of the memory map can be found in later sections of this manual, see Table 4-2.

Figure 4-2: *MPC8548 Memory Map*

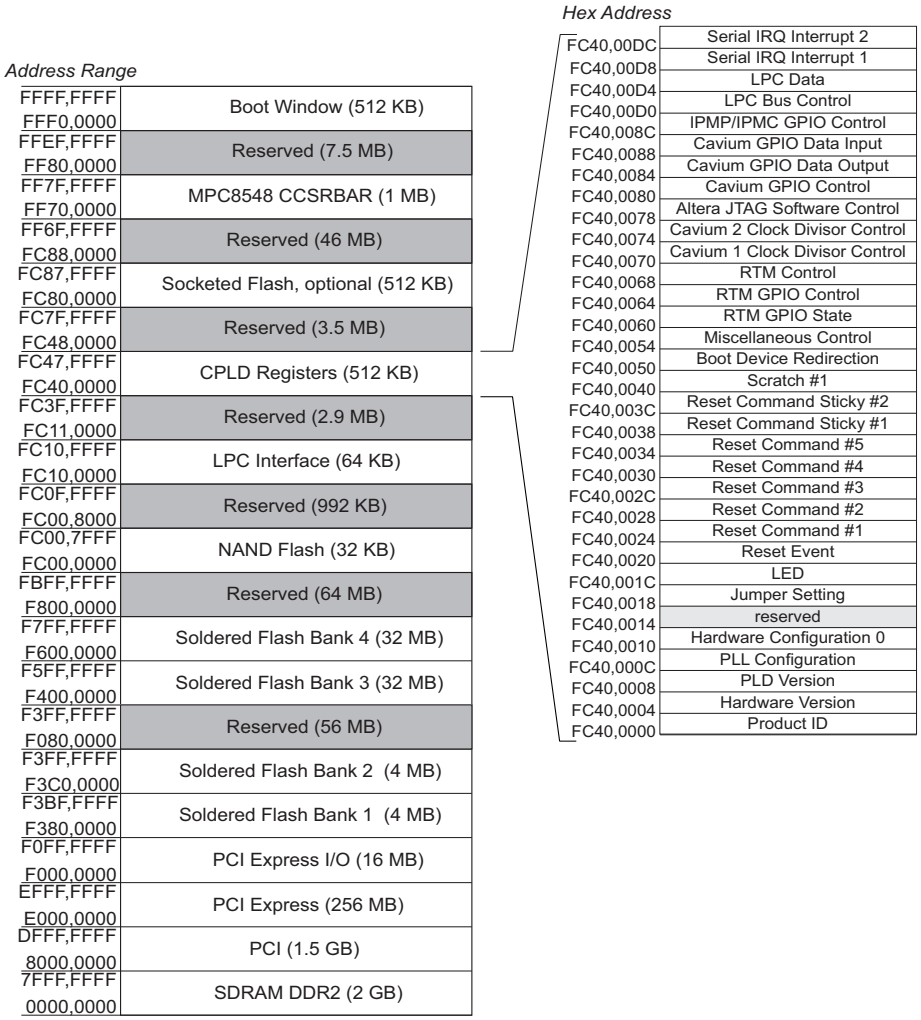


Table 4-2: *MPC8548 Address Summary*

Hex Physical Address:	Access Mode:	Register Description:	See Page:
FFF8,0000	R/W	Boot window (512 KB)	–
FF80.0000	–	reserved (7.5 MB)	
FF70,0000	R/W	MPC8548 CCSRBAR (1MB)	–
FC88,0000	–	reserved (46 MB)	

Hex Physical Address:	Access Mode:	Register Description: (continued)	See Page:
FC80,0000	R/W	Socketed flash, optional (512 KB)	4-7
FC48,0000	–	reserved (3.5 MB)	
FC40,00DC0	R/W	Serial IRQ Interrupt 2	5-15
FC40,00D8	R/W	Serial IRQ Interrupt 1	5-15
FC40,00D4	R/W	LPC Data	5-15
FC40,00D0	R/W	Low Pin Count (LPC) Bus Control	5-14
FC40,008C	R/W	IPMP/IPMC GPIO Control	5-14
FC40,0088	R/W	Cavium GPIO Data Input	5-13
FC40,0084	R/W	Cavium GPIO Data Output	5-13
FC40,0080	R/W	Cavium GPIO Control	5-12
FC40,0078	R/W	Altera JTAG Chain Software Control	5-12
FC40,0074	R/W	Cavium 2 C_MUL Clock Divisor Control	5-11
FC40,0070	R/W	Cavium 1 C_MUL Clock Divisor Control	5-11
FC40,0068	R/W	RTM Control	5-10
FC40,0064	R/W	RTM GPIO Control	5-10
FC40,0060	R/W	RTM GPIO State	5-10
FC40,0054	R/W	Miscellaneous Control (SIO, I2C, Test Clock)	5-9
FC40,0050	R/W	Boot Device Redirection	5-8
FC40,0040	R/W	Scratch #1	–
FC40,003C	R/W	Reset Command Sticky #2	5-8
FC40,0038	R/W	Reset Command Sticky #1	5-7
FC40,0034	W	Reset Command #5	5-7
FC40,0030	W	Reset Command #4	5-7
FC40,002C	W	Reset Command #3	5-6
FC40,0028	W	Reset Command #2	5-6
FC40,0024	W	Reset Command #1	5-5
FC40,0020	R/W	Reset Event	5-5
FC40,001C	R/W	LED	5-4
FC40,0018	R/W	Jumper Setting	5-4
FC40,0014	–	reserved	–
FC40,0010	R/W	Hardware Configuration 0	5-3
FC40,000C	R/W	PLL Configuration	5-3
FC40,0008	R/W	PLD Version	5-3
FC40,0004	R/W	Hardware Version	5-2
FC40,0000	R/W	Product ID (CPLD 512 KB)	5-2
FC11,0000		reserved (2.9 MB)	
FC10,0000	R/W	LPC Interface (64 KB)	4-5
FC00,8000	–	reserved (992 KB)	
FC00,0000	R/W	NAND flash (32 KB)	4-8

Hex Physical Address:	Access Mode:	Register Description: (continued)	See Page:
F800,0000	–	reserved (64 MB)	
F600,0000	R/W	Soldered flash bank 4 (32 MB)	4-7
F400,0000	R/W	Soldered flash bank 3 (32 MB)	4-7
F080,0000	–	reserved (56 MB)	
F3C0,0000	R/W	Soldered flash bank 2 (4 MB)	4-7
F380,0000	R/W	Soldered flash bank 1 (4 MB)	4-7
F000,0000	R/W	PCI Express I/O space (16 MB)	4-8
E000,0000	R/W	PCI Express (256 MB)	4-8
8000,0000	R/W	PCI (1.5 GB)	4-8
0000,0000	R/W	SDRAM DDR2 (2 GB)	4-7

Chip Selects

The MPC8548 memory controller functions as a chip select (CS) generator to access on-board memory devices. In order to select one device over another, the following chip selects have been established.

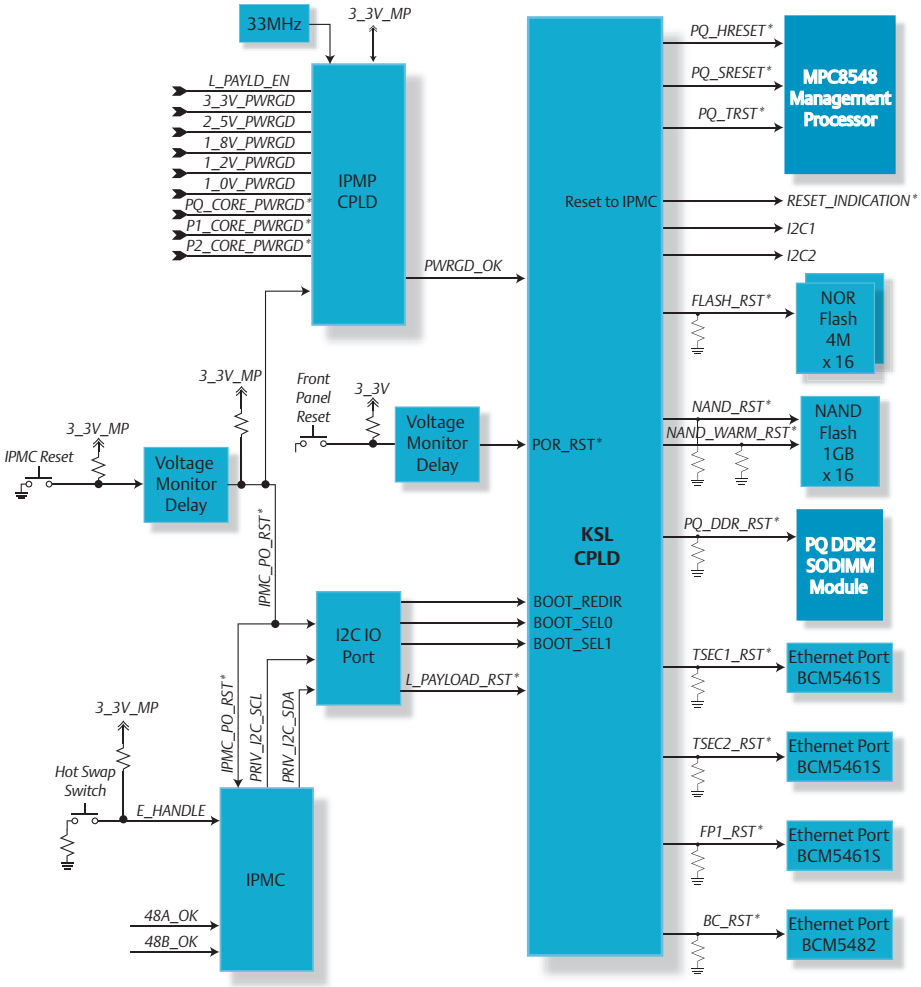
Table 4-3: *Device Chip Selects*

Pin:	Signal:
0	Boot bank [1]
1	Soldered flash boot bank 1 (default)
2	Soldered flash boot bank 2
3	Socketed flash (optional)
4	KSL CPLD registers
5	NAND flash
6	Soldered NOR flash boot banks 3 and 4
7	LPC interface

1. Boot bank can be either socketed flash, flash 1, or flash 2; depending on the jumper setting (see Fig. 2-6).

Reset Diagram

Figure 4-3: *MPC8548 Reset Diagram*



MEMORY

The memory devices in the management complex consist of:

- 1 GB DDR2 SDRAM
- 512 KB socketed flash
- 8 MB soldered NOR flash (two redundant banks of 4 MB each)
- 1 GB soldered NAND flash
- 512 Mb or 64 MB soldered NOR flash

SDRAM

This is a specialized, socketed, 200-pin, small outline, clocked, dual in- line, memory module (SO-CDIMM). It provides Error-correcting Code (ECC) on the SDRAM memory bus operating at 200 MHz. The MPC8548 detects all double-bit errors, multi-bit errors within a nibble and corrects all single-bit errors.

The 128M X 72 DDR2 SDRAM is a high-density, un-buffered SO-CDIMM. This module consists of nine 128x8-bit with eight banks DDR2 SDRAMs, a zero delay phase-lock loop (PLL) clock, and a 2 KB serial presence detect (SPD) EEPROM. The SDRAM starts at physical address $0000{,}0000_{16}$.

Flash

There are several flash devices on the local bus interfacing the CPLD and MPC8548 processor. The four soldered flash banks are labeled 1 through 4:

- Banks 1 and 2 are the MPC8548 U-boot banks (see “4M x 16”). These boot banks are used in the boot redirection scheme, see “Boot Device Redirection (BDR).”
- Banks 3 and 4 are physically one device, but appear in the software as two banks of 32 MB (see “64 MB x 16”). These are for general purpose storage.

512 KB x 8 (optional)

The 512 KB of 32-pin PLCC socketed flash starts at physical address $FC80{,}0000_{16}$ and is used for Engineering code. The StrataFlash (P33) features high-performance fast asynchronous access times, low power, and flexible security options.

4M x 16

The two 4 MB soldered flash devices are used for MPC8548 boot code. This redundant bank configuration allows booting from either bank in case of corruption in one bank. See “Boot Device Redirection (BDR)” on page 7-41. The SST NOR flash devices are organized as 4Mx8

in a dual-bank architecture for concurrent read/write operation with hardware and software data protection schemes. These devices start at physical addresses $F000,0000_{16}$ (boot bank 1) and $F040,0000_{16}$ (boot bank 2).

1 GB x 16

The ATCA-9305 uses 1 GB of M-Systems DiskOnChip (mDOC H3) NAND flash starting at physical address $FC00,0000_{16}$ for non-volatile RAM storage and True Flash File System (TFFS). This memory incorporates an embedded flash controller and memory, and includes hardware protection and security-enabling features, an enhanced programmable boot block enabling eXecution In Place (XIP) functionality using 16-bit access, user-controlled One Time Programmable (OTP) partitions, and 6-bit Error Detection Code/Error Correction Code (EDC/ECC).

64 MB x 16

The 64 MB soldered NOR flash starts at physical address $F400,0000_{16}$ (bank 3). The 64-Mbit P33 device provides CN5860 code storage and non-volatile memory.

PCI

The MPC8548 performs all the functions of a PCI host and monarch, and handles all arbitration and enumeration functions. PCI starts at physical address $8000,0000_{16}$.

The PCI bus connects to both Cavium processors, the MPC8548 processor and the Broadcom Ethernet switch, see Table 4-4. All of the devices on the PCI bus can operate at 66 MHz and perform 64-bit transactions in conventional PCI mode except for the Broadcom switch. The switch has a 32-bit PCI bus.

The MPC8548 stores the Cavium CN5860 operating system and monitor code in local memory and then uses the boot over PCI functionality to bring up the CN5860 processor complexes.

Table 4-4: *PCI Device Interrupts and ID Assignments*

PCI Device:	Interrupt:	IDSEL:
Cavium processor 1	IRQ6	PCI_AD11
Cavium processor 2	IRQ5	PCI_AD12
Ethernet switch	IRQ4	PCI_AD13
MPC8548	–	PCI_AD14

PCI Express

The four lane PCIe routes between the MPC8548 and the optional rear transition module (zone 3 connector). PCIe starts at physical address $E000,0000_{16}$.

I²C INTERFACE

The I²C interface consists of the MPC8548 initialization EEPROM, user (storage) NVRAM, SO-CDIMM, and the Real-time Clock (RTC). The two Atmel two-wire serial EEPROMs on the I²C interface consist of the Serial Clock (SCL) input and the Serial Data (SDA) bidirectional lines.

Table 4-5: *I²C Device Addresses*

I²C Device:	Address:
MPC8548 Initialization (EEPROM-2)	0xA0
User NVRAM (EEPROM-1)	0xA2
DDR2 SDRAM (SO-CDIMM)	0xA4
M41T00 RTC	0xD0

The two EEPROMs store non-volatile information such as board, monitor, and operating system configurations as well as customer specific items.

Table 4-6: *MPC8548 NVRAM Memory Map*

EEPROM:	Address Offset (hex):	Description:	Window Size (bytes)
EEPROM-1 0xA2 (write protected)	0x1FF0-0x1FFF	Boot verify secondary area (monitor)	16
	0x1FE0-0x1FEF	Boot verify primary area (monitor)	16
	0x1EE0-0x1FDF	Operating system parameters (monitor)	256
	0x0000-x1EDF	User defined	7903
EEPROM-2 0xA0 (write protected)	0x0900-0x1FFF	Emerson reserved area	5887
	0x0800-0x08FF	Miscellaneous	256
	0x07F0-0x07FF	Power-on Self-test (POST)	16
	0x0000-0x07EF	User defined	2032

Note: *Both EEPROMs are write-protected.*

MANAGEMENT PROCESSOR HEADER AND SERIAL PORT

JTAG/COP Interface (optional)

The management complex uses header P2 for debug purposes.

Table 4-7: *Serial Debug Connector, P2*

Pin:	Signal:	Description:
1	PQ_TDO	Test Data Output is the serial data output as well as test and programming data.
2	no connect	–

Pin:	Signal:	Description: (continued)
3	PQ_TDI	Test Data Input is the serial input pin for instructions as well as test and programming data.
4	DEBUG_TRST*	Test Reset input signal resets the test access port.
5	no connect	–
6	PQ_JTAG_PWR	3.3 volt power
7	PQ_TCK_R	Test Clock Input is the clock input to the boundary scan test (BST) circuitry.
8	no connect	–
9	PQ_TMS	Test Mode Select input pin provides the control signal to determine the transitions of the TAP controller state machine.
10	no connect	–
11	DEBUG_SRESET*	Soft Reset input signal indicates that the MPC8548 must initiate a System Reset interrupt.
12	ground	–
13	DEBUG_HRESET*	Hard Reset input signal indicates that a complete Power-on Reset must be initiated by the MPC8548.
14	no connect	–
15	PQ_CKSTP_OUT*	Checkstop Out indicates the MPC8548 has detected a checkstop condition and has ceased operation.
16	ground	–

Serial Debug Port

The console port for the management processor is accessible via the front panel mini-B USB connector P7. The supported baud rates for these ports operate at 9600, 14400, 19200, 38400, 57600, and 115200 bps.

Table 4-8: *Serial Debug Connector, P7*

Pin:	Signal:
1	no connect
2	PQ_CONSOLE_RX_C
3	PQ_CONSOLE_TX_C
4	no connect
5	signal ground
6	chassis ground
7	chassis ground

Management Processor CPLD

The ATCA-9305 uses a Programmable Logic Device (PLD) to provide control logic for the local bus. The PLD implements various registers for reset, hardware, and LPC bus communication between the processors.

MPC8548 PLD REGISTER SUMMARY

The PLD registers start at address FC40,0000$_{16}$. As a rule, registers retain their values through all resets except for power-on and front panel reset. Table 5-1 lists the 8-bit PLD registers followed by the register bit descriptions.

Table 5-1: *PLD Register Summary*

Address Offset (hex):	Mnemonic:	Register Name:	See Page:
0x00	PIDR	Product ID	5-2
0x04	HVR	Hardware Version	5-2
0x08	PVR	PLD Version	5-3
0x0C	PLLCR	PLL Configuration	5-3
0x10	HCR00	Hardware Configuration 0	5-4
0x18	JSR	Jumper Setting	5-4
0x1C	LEDR	LED	5-5
0x20	RER	Reset Event	5-5
0x24	RCR1	Reset Command #1	5-6
0x28	RCR2	Reset Command #2	5-6
0x2C	RCR3	Reset Command #3	5-6
0x30	RCR4	Reset Command #4	5-7
0x34	RCR5	Reset Command #5	5-7
0x38	RCRS1	Reset Command Sticky #1	5-8
0x3C	RCRS2	Reset Command Sticky #2	5-8
0x40	SCR1	Scratch #1[1]	–
0x50	BDRR	Boot Device Redirection	5-9
0x54	MISC	Miscellaneous Control (SIO, I2C, Test Clock)	5-9
0x58	LFTR1	Low Frequency Timer 1	5-9
0x5C	LFTR2	Low Frequency Timer 2	5-9
0x60	RGSR	RTM GPIO State	5-10
0x64	RGCR	RTM GPIO Control	5-10
0x68	RTMCR	RTM Control	5-11
0x70	CMUL1	Cavium 1 C_MUL Clock Divisor Control	5-11
0x74	CMUL2	Cavium 2 C_MUL Clock Divisor Control	5-12
0x78	JTAG	Altera JTAG Chain Software Control	5-12
0x80	CGCR	Cavium GPIO Control	5-12

Address Offset (hex):	Mnemonic:	Register Name: (continued)	See Page:
0x84	CGDO	Cavium GPIO Data Out	5-13
0x88	CGDI	Cavium GPIO Data In	5-13
0x8C	IGCR	IPMP/IPMC GPIO Control	5-14
0xD0	LPC1	Low Pin Count (LPC) Bus Control	5-14
0xD4	LPCD	LPC Data	5-15
0xD8	SIRQI1	Serial IRQ Interrupt 1 [15:8]	5-15
0xDC	SIRQI2	Serial IRQ Interrupt 2 [7:0]	5-15

1. Scratch 1 (0x40) is a read/write register for storage only.

Product ID

This read-only register identifies the board as ATCA-9305, and is used for PLD coding.

Register 5-1: *Product ID (0x00)*

Bits:	Function:	Description:
7	CAVF1	Cavium Frequency 1
6	CAVF0	Cavium Frequency 0
5	0	Product ID
4	0	
3	0	
2	0	
1	HC1	Hardware Configuration 1
0	HC0	Hardware Configuration 0

Hardware Version

This read-only register tracks hardware revisions.

Register 5-2: *Hardware Version (0x04)*

Bits:	Function:	Description:
7	0	
6	0	
5	0	
4	0	
3	HVN (3)	Hardware Version Number is hard coded in the PLD and changes with every major PCB artwork version. Version starts at 00_{16}.
2	HVN (2)	
1	HVN (1)	
0	HVN (0)	

PLD Version

This read-only register tracks PLD revisions.

Register 5-3: *PLD Version (0x08)*

Bits:	Function:	Description:
7	0	This is hard coded in the PLD and changes with every released code change. Version starts at 00_{16}.
6	0	
5	0	
4	0	
3	0	
2	0	
1	0	
0	0	

PLL Reset Configuration

Write to this register to reconfigure the SYSCLK to CCB clock ratio and the CCB to CORE clock ratio using valid values from the *MPC8548E PowerQUICC™ III Integrated Processor Family Reference Manual*. The changes take affect when the processor is reset (for example, the software hard reset command or watchdog timer expires). Default values are restored when the board is power-cycled, front panel reset is pressed, or receives a PCI reset that was not the result of the MPC8548 software initiating a PCI RSTOUT command.

Register 5-4: *PLL Reset Configuration (0x0C)*

Bits:	Function:	Description:
7	reserved	
6	CCCB2	CCB2 to CORE clock ratio
5	CCCB1	CCB1 to CORE clock ratio
4	CCCB0	CCB0 to CORE clock ratio
3	CCBSYS3	SYSCLOCK3 to CCB clock ratio
2	CCBSYS2	SYSCLOCK2 to CCB clock ratio
1	CCBSYS1	SYSCLOCK1 to CCB clock ratio
0	CCBSYS0	SYSCLOCK0 to CCB clock ratio

Hardware Configuration 0

The read-only HCR0 allows the MPC8548 monitor software to easily determine specific hardware configurations, such as the processor clock and MPC8548 DDR memory.

Register 5-5: *Hardware Configuration 0 (0x10)*

Bits:	Function:	Description:
7	0	
6	P33P	P33 (StrataFlash) is Present
5	RST_IND_CLR	Clear the Reset Indication to the IPMC controller
4	CAVF1	Cavium Frequency 1
3	CAVF0	Cavium Frequency 0
2	PQCF1	MPC8548 Core Frequency 1
1	PQCF0	MPC8548 Core Frequency 0
0	PQDDRF	MPC8548 DDR SDRAM Fast

Jumper Settings

These read-only bits may be read by software to determine the current jumper settings. See the jumper descriptions on page 2-8.

Register 5-6: *Jumper Settings (0x18)*

Bits:	Function:	Description:
7	0	
6	0	
5	0	
4	SJ	Cavium Boot Flash Jumper 0 Installed, Cavium processors boot from soldered flash 1 Not installed, Cavium processors boot from socket
3	BOOT	Boot PCI Jumper 0 Installed, boot from flash (socket or soldered per bit 4) 1 Not installed, boot over PCI from the MPC8548
2	REDIR	Boot Redirect Jumper 0 Installed, disables boot redirection 1 Not installed, enables boot redirection
1	IG ROM	Ignore SROM 0 Not installed, SROM is used for initialization (default) 1 Installed, disables SROM, uses default values in monitor code
0	BT SKT	Boot from Socket 0 Not installed, enables MPC8548 to boot from soldered flash (default) 1 Installed, enables MPC8548 to boot from socketed flash

LED

Writing a one to an LED bit lights that LED. During monitor power-up, the debug LEDs are used to display the software progress.

Register 5-7: *LED (0x1C)*

Bits:	Function:	Description:
7	PQRED	MPC8548 red LED Lit on power-up and turned off when the monitor finishes boot up and Power-on Self Testing (POST)
6	PQGREEN	MPC8548 green LED
5	SWLEDCLK	Ethernet Switch LED Clock
4	SWLEDDAT	Ethernet Switch LED Data
3	DEBUGLED3	LED CR22
2	DEBUGLED2	LED CR21
1	DEBUGLED1	LED CR19
0	DEBUGLED0	LED CR18

Reset Event

This read-only register contains the bit corresponding to the most recent event which caused a reset. When power is first applied, the FP_PSH_BUTTN reset event is not latched into the Reset Event register, this is the Power-on Reset (POR) event. Front panel reset events which occur after power-up will be latched.

Note: *At power-up, the FRST_PWR_UP defaults to 1.*

Register 5-8: *Reset Event (0x20)*

Bits:	Function:	Description:
7	RTMPB	RTM push button
6	SHR	Software Hard Reset Set to 1 when the last reset was caused by a write to the Reset Command register
5	CPUHRR	CPU Hard Reset Request
4	COPSR	Set to 1 when a COP header or software-issued Soft Reset (SRESET) has occurred
3	COPHR	Set to 1 when a COP header Hard Reset (HRESET) has occurred
2	PAYR	Set to 1 when a Payload Reset from the IPMC has occurred
1	SBR	Software Board Reset Set to 1 when the IPMC software issued the board (payload) reset
0	FPPB	Front Panel Push Button (FP_PSH_BUTTN, POR_RST)

Reset Command 1

The write-only Reset Command 1 register forces one of several types of resets, as shown below. A reset sequence is first initiated by writing a one to a single valid bit, then the PLD performs that particular reset, and the bit is automatically cleared.

Register 5-9: *Reset Command 1 (0x24)*

Bits:	Function:	Description:
7	WBR	Reset the Whole Board
6	PQCR	Reset the MPC8548 Complex
5	CAV1CR	Reset the Cavium CN5860 1 Complex
4	CAV2CR	Reset the Cavium CN5860 2 Complex
3	SWICR	Reset the switch BCM5680x Complex
2	I2C R	Reset the I2C on the MPC8548
1	RTMR	Reset the (optional) RTM
0	reserved	

Reset Command 2

The write-only Reset Command 2 register forces one of several types of MPC8548 resets, as shown below. A reset sequence is first initiated by writing a one to a single valid bit, then the PLD performs that particular reset, and the bit is automatically cleared.

Register 5-10: *Reset Command 2 (0x28)*

Bits:	Function:	Description:
7	PQHR	MPC8548 Hardware Reset
6	PQSR	MPC8548 Software Reset
5	PQDR	MPC8548 DDR SDRAM Reset
4	PQF	MPC8548 Flash reset
3	NANDR	MPC8548 NAND flash Reset
2	NANDWR	MPC8548 NAND flash Warm Reset
1	reserved	
0	reserved	

Reset Command 3

The write-only Reset Command 3 register forces one of several types of Cavium 1 resets, as shown below. A reset sequence is first initiated by writing a one to a single valid bit, then the PLD performs that particular reset, and the bit is automatically cleared.

Register 5-11: *Reset Command 3 (0x2C)*

Bits:	Function:	Description:
7	CAV1R	Cavium 1 Reset
6	CAV1PR	Cavium 1 PCI Reset
5	CAV1DR	Cavium 1 DDR SDRAM Reset
4	CAV1F	Cavium 1 4 MB Flash (Cavium local bus) reset
3	CAV1M1	Cavium 1 MIP1 reset
2	CAV1M2	Cavium 1 MIP2 reset

Bits:	Function:	Description: (continued)
1	reserved	
0	reserved	

Reset Command 4

The write-only Reset Command 4 register forces one of several types of Cavium 2 resets, as shown below. A reset sequence is first initiated by writing a one to a single valid bit, then the PLD performs that particular reset, and the bit is automatically cleared.

Register 5-12: *Reset Command 4 (0x30)*

Bits:	Function:	Description:
7	CAV2R	Cavium 2 Reset
6	CAV2PR	Cavium 2 PCI Reset
5	CAV2DR	Cavium 2 DDR SDRAM Reset
4	CAV2F	Cavium 2 4 MB Flash (Cavium local bus) reset
3	CAV2M3	Cavium 2 MIP3 reset
2	CAV2M4	Cavium 2 MIP4 reset
1	reserved	
0	reserved	

Reset Command 5

The write-only Reset Command 5 register forces one of several types of BCM5680x Ethernet switch resets, as shown below. A reset sequence is first initiated by writing a one to a single valid bit, then the PLD performs that particular reset, and the bit is automatically cleared.

Register 5-13: *Reset Command 5 (0x34)*

Bits:	Function:	Description:
7	SWIR	Switch Reset
6	TSEC1R	TSEC1 Ethernet to front panel PHY Reset
5	TSEC2R	TSEC2 Ethernet to switch PHY Reset
4	FPIR	FPI Ethernet to front panel PHY Reset
3	BCR	Ethernet dual PHY to backplane Base Channel reset
2	reserved	
1	reserved	
0	reserved	

Reset Command Sticky #1

The read/write Reset Command Sticky #1 register forces one of several types of the group-complex resets, as shown below. A reset sequence is first initiated by writing a one to one or more bits, then the PLD performs that particular reset. The bit will persist until cleared.

Note: *The board powers down and powers back up when the Cavium processors power is back up (bits 0 or 1 are cleared).*

Register 5-14: *Reset Command Sticky #1 (0x38)*

Bits:	Function:	Description:
7	CAV1C	Cavium 1 Complex reset
6	CAV2C	Cavium 2 Complex reset
5	SWIC	Switch Complex reset
4	CAV1CF	Cavium 1 Complex 4MB Flash reset
3	CAV2CF	Cavium 2 Complex 4MB Flash reset
2	NANDF	NAND Flash reset
1	CAV2RPD	Reset and power down the Cavium 2 core
0	CAV1RPD	Reset and power down the Cavium 1 core

Reset Command Sticky #2

The read/write Reset Command Sticky #2 register forces one of several types of the PHY reset command, as shown below. A reset sequence is first initiated by writing a one to one or more bits, then the PLD performs that particular reset. The bit will persist until cleared.

Register 5-15: *Reset Command Sticky #2 (0x3C)*

Bits:	Function:	Description:
7	TSEC1R	TSEC1 Ethernet to front panel PHY Reset
6	TSEC2R	TSEC2 Ethernet to switch PHY Reset
5	FPIR	FPI Ethernet from switch to front panel PHY Reset
4	BCR	Ethernet dual PHY to backplane Base Channel Reset
3	MIP1	SPI to XAUI bridge #1 on Cavium 1
2	MIP2	SPI to XAUI bridge #2 on Cavium 1
1	MIP3	SPI to XAUI bridge #3 on Cavium 2
0	MIP4	SPI to XAUI bridge #4 on Cavium 2

Boot Device Redirection

The read/write Boot Device Redirection register (BDRR) allows the user to determine which of three boot devices the MPC8548 CPU is using as the boot device. Several bits also indicate which device was set as the initial boot device. The Boot Redirected bit is set to a 1 when the current boot device does not match the initial default boot device. This indicates to the user that the image in the default device was bad, the MPC8548 watch dog timer expired, and the next device was tried. The boot device redirection order is determined by IPMC. Reference the “Boot Device Diagram”.

Register 5-16: *Boot Device Redirection (0x50)*

Bits:	Function:	Description:
7	SELFRS	Self Refresh Started
6	BOOTSEL1	IPMC successful boot indication (BOARD_BOOTED)
5	reserved	
4	BSJ	Boot from Socket Jumper A shunt on J9 [1:2] selects the 512KB socketed ROM as the boot device, see Fig. 2-6.
3	NFBS	Nand Flash Busy Signal
2	BDS	Active boot device is socket
1	BDF1	Active boot device is flash 2
0	BDF0	Active boot device is flash 1

Miscellaneous Control

This register includes two bits for manually toggling the MPC8548 I^2C bus.

Register 5-17: *Miscellaneous Control (0x54)*

Bits:	Function:	Description:
7	P33WP	0 Write Protect disabled (default until the monitor boots) 1 Write Protect enabled
6	SROM1WP	0 Write Protect disabled 1 Write Protect enabled (default)
5	SROM0WP	0 Write Protect disabled 1 Write Protect enabled (default)
4	FLASH1WP	0 Write Protect disabled (default until the monitor boots) 1 Write Protect enabled
3	FLASH0WP	0 Write Protect disabled (default until the monitor boots) 1 Write Protect enabled
2	NANDWP	0 Write Protect disabled 1 Write Protect enabled (default)
1	I2CSDA	I^2C Data line 0 Drive a 0 onto the I2C SDA line 1 Drive a 1 onto the I2C SDA line
0	I2CSCL	I^2C Clock line 0 Drive a 0 onto the I2C SCL line 1 Drive a 1 onto the I2C SCL line

Low Frequency Timer 1 and 2

Registers LFTR1 (0x58) and LFTR2 (0x5C) are timers. They determine how many 50 µs intervals you want before the next interrupt on Cavium GPIO5.

Note: *Unless the frequency is set to 0, there is always one 50 µs interval. This is the reason for the register setting being 1 less than an even hundred, for example 199 rather than 200.*

Table 5-2: *Low Frequency Timer Settings*

Frequency:	Set Register:	Comments:
0	Off	Never interrupts
1 Hz	19999 (0x4E1F)	These frequencies require the use of both registers
10 Hz	1999 (0x7CF)	
100 Hz	199 (0xC7)	
1 KHz	19 (0x13)	
10 KHz	1	This equals two 50 µs time units (default)

RTM GPIO State

This read-only register reads the current state of the GPIO pins.

Register 5-18: *RTM GPIO State (0x60)*

Bits:	Function:	Description:
7	RTM_GPIO 7	
6	RTM_GPIO 6	
5	RTM_GPIO 5	
4	RTM_GPIO 4	
3	RTM_GPIO 3	
2	RTM_GPIO 2	
1	RTM_GPIO 1	
0	RTM_GPIO 0	

RTM GPIO Control

This register sets the state of the GPIO pins. These signals are implemented as open collector signals.

Register 5-19: *RTM GPIO Control (0x64)*

Bits:	Function:	Description:
7	RTM_GPIO 7	0 Causes the corresponding bit to be driven to 0 1 Tristates the signal; this will either be read by the RTM as a 1 or can be driven by the RTM to any value
6	RTM_GPIO 6	
5	RTM_GPIO 5	
4	RTM_GPIO 4	
3	RTM_GPIO 3	
2	RTM_GPIO 2	
1	RTM_GPIO 1	
0	RTM_GPIO 0	

RTM Status

The RTM identification (ID) is determined by factory installed configuration resistors.

Register 5-20: *RTM Control (0x68)*

Bits:	Function:	Description:
7	0	
6	0	
5	0	
4	RTMP	RTM is Present
3	RTMID3	RTM Identification bits 3:0
2	RTMID2	0000 = Test RTM (factory only) 1000 = 20GbE I/O RTM 1100 = 18GbE and 2x10GbE I/O RTM 1010 = Storage RTM
1	RTMID1	
0	RTMID0	

Cavium 1 C_MUL Clock Divisor Control

Use the C_MUL1 register to reduce the speed of the Cavium CN5860 processor 1 core.

Caution: **Do not over-clock the Cavium frequency (bits 6:7 hard strapped).**

Register 5-21: *Cavium 1 C_MULL Clock Divisor Control (0x70)*

Bits:	Function:	Description:
7	CAVF	Cavium Frequency resistor set bit (read-only) 00 600 01 750 10 800 11 reserved
6		
5	CMULOE	C_MUL Output Enable
4	P1CMUL4	These bits drive directly to the Cavium 1. The core clock speed is the number multiplied by 50 MHz. For example, the 800 MHz core is set to 16(0x10).
3	P1CMUL3	
2	P1CMUL2	
1	P1CMUL1	
0	P1CMUL0	

Cavium 2 C_MUL Clock Divisor Control

Use the C_MUL2 register to reduce the speed of the Cavium CN5860 processor 2 core.

Caution: **Do not over-clock the Cavium frequency (bits 6:7 hard strapped).**

Register 5-22: *Cavium 2 C_MULL Clock Divisor Control (0x74)*

Bits:	Function:	Description:
7	CAVF1	Cavium 1 Frequency resistor set bit (read-only, see Register Map 5-21)
6	CAVF0	Cavium 0 Frequency resistor set bit (read-only)
5	CMULOE	C_MUL Output Enable
4	P1CMUL4	These bits drive directly to the Cavium 2. The core clock speed is the number multiplied by 50 MHz. For example, the 800 MHz core is set to 16(0x10).
3	P1CMUL3	
2	P1CMUL2	
1	P1CMUL1	
0	P1CMUL0	

JTAG

This register allows for manual reprogramming of the PLDs on the board. Changes to this register do not take effect until after a full board reset.

Register 5-23: *JTAG (0x78)*

Bits:	Function:	Description:
7	reserved	
6	reserved	
5	JTAGOEN	JTAG Output Enable
4	JTAGTCKSEL	JTAG Test Clock Select changes from header to PLD as the TCK source
3	JTAGTCK	JTAG Test Clock
2	JTAGTMS	JTAG Test Mode Select
1	JTAGTDO	JTAG Test Data Output
0	JTAGTDI	JTAG Test Data Input (read only)

Cavium GPIO Control

Each Cavium processor has three GPIO control bits connected to the PLD. This register determines whether the PLD is driving or receiving on these lines. Setting a bit to 1 causes the PLD to drive the corresponding line.

Register 5-24: *Cavium GPIO Control (0x80)*

Bits:	Function:	Description:
7	reserved	
6	reserved	
5	P2GPIO5OE	Processor 2 GPIO5 Output Enable (enabled is the default) Output enable is set for the TIC timer output to the Cavium
4	P2GPIO4OE	Processor 2 GPIO4 Output Enable This is an input from the Cavium to reset the MIP4

Bits:	Function:	Description: (continued)
3	P2GPIO3OE	Processor 2 GPIO3 Output Enable This is an input from the Cavium to reset the MIP3
2	P1GPIO5OE	Processor 1 GPIO5 Output Enable (enabled is the default) Output enable is set for the TIC timer output to the Cavium
1	P1GPIO4OE	Processor 1 GPIO4 Output Enable This is an input from the Cavium to reset the MIP2
0	P1GPIO3OE	Processor 1 GPIO3 Output Enable This is an input from the Cavium to reset the MIP1

Cavium GPIO Data Out

This register is the data that will be driven on the GPIO line when the Output enable is set.

Register 5-25: *Cavium GPIO Data Out (0x84)*

Bits:	Function:	Description:
7	reserved	
6	reserved	
5	reserved	
4	P2GPIO4	Set the value of the Cavium 2 GPIO bit 4
3	P2GPIO3	Set the value of the Cavium 2 GPIO bit 3
2	reserved	
1	P1GPIO4	Set the value of the Cavium 1 GPIO bit 4
0	P1GPIO3	Set the value of the Cavium 1 GPIO bit 3

Cavium GPIO Data In

This register reads the value on the GPIO lines connected to each Cavium.

Register 5-26: *Cavium GPIO Data In (0x88)*

Bits:	Function:	Description:
7	reserved	
6	reserved	
5	reserved	
4	P2GPIO4	Read the value of the Cavium 2 GPIO bit 4
3	P2GPIO3	Read the value of the Cavium 2 GPIO bit 3
2	reserved	
1	P1GPIO4	Read the value of the Cavium 1 GPIO bit 4
0	P1GPIO3	Read the value of the Cavium 1 GPIO bit 3

IPMP/IPMC GPIO Control

This register provides access (if required) to signals between the KSL CPLD and the IPMP, as well as to signals between the KSL CPLD and the IPMC. The lower two bits can request request the power down of a Cavium core from the sticky reset register.

Register 5-27: *IPMP/IPMC GPIO Control (0x8C)*

Bits:	Function:	Description:
7	IPMC2KSL4	Input only
6	IPMC2KSL3	
5	IPMC2KSL2	
4	IPMC2KSL1	
3	IPMP2KSL4	Output only
2	IPMP2KSL3	Output only
1	IPMP2KSL2	Power-down signal for Cavium 2 (output) Assert high to shut down the core. The sticky Cavium reset also causes this to be asserted.
0	IPMP2KSL1	Power-down signal for Cavium 1 (output) Assert high to shut down the core. The sticky Cavium reset also causes this to be asserted.

LPC Bus Control

This is the control register for the 4-bit LPC bus. It allows for communication with the IPMC controller from the management CPU.

Register 5-28: *LPC Bus (0xD0)*

Bits:	Function:	Description:
7	LPCIE	LPC Interrupt Enable
6	LPCS	LPC State (internal use only)
5		
4		
3		
2	LPCIOE	LPC I/O Error
1	SYNCE	SYNC Error
0	SYNCT	SYNC Time-out

LPC Data

This is the data register for the 4-bit LPC bus. It allows for communication with the IPMC controller from the management CPU. This register provides the data to be sent or received, depending upon the commands given in the control register.

Register 5-29: *LPC Data (0xD4)*

Bits:	Function:	Description:
7:0	-	LPC Data

Serial IRQ Interrupt 1

This is interrupt register1 for the LPC bus.

Register 5-30: *Serial IRQ Interrupts 1 (0xD8)*

Bits:	Function:	Description:
7:0	-	Interrupts

Serial IRQ Interrupt 2

This is interrupt register2 for the LPC bus.

Register 5-31: *Serial IRQ Interrupts 2 (0xDC)*

Bits:	Function:	Description:
7:0	-	Interrupts

(blank page)

Ethernet Interface

The ATCA-9305 supports multiple Ethernet interfaces. This chapter describes the Broadcom BCM56802 switch, PHYS BCM5482 and BCM5461S, Ethernet address, LEDs and connectors.

BROADCOM BCM56802 SWITCH

The BCM56802 is a 16-port, 10-GbE multi-layer switch based on the StrataXGS® architecture. The switch operates at 66 MHz with a 32-bit PCI bus for processor communication. SERDES functionality includes 10-Gbps XAUI and 1-Gbps SGMII PHY interfaces.

One 10/100/1000BASE-T Ethernet (SGMII) port is routed to a front panel RJ45 connector (see Fig. 6-1), one is routed to the MPC8548 management processor TSEC2 port, and two are routed to the base channel backplane (see Fig. 8-2). Two 10 GbE XAUI ports connect to the back panel via the fabric channel (see Fig. 8-2).

Two XAUI ports process packets to and from each CN5860 processor. Six 10 GbE XAUI ports route to the optional rear transition module (RTM). See Table 8-3 and Table 8-4 for pin assignments.

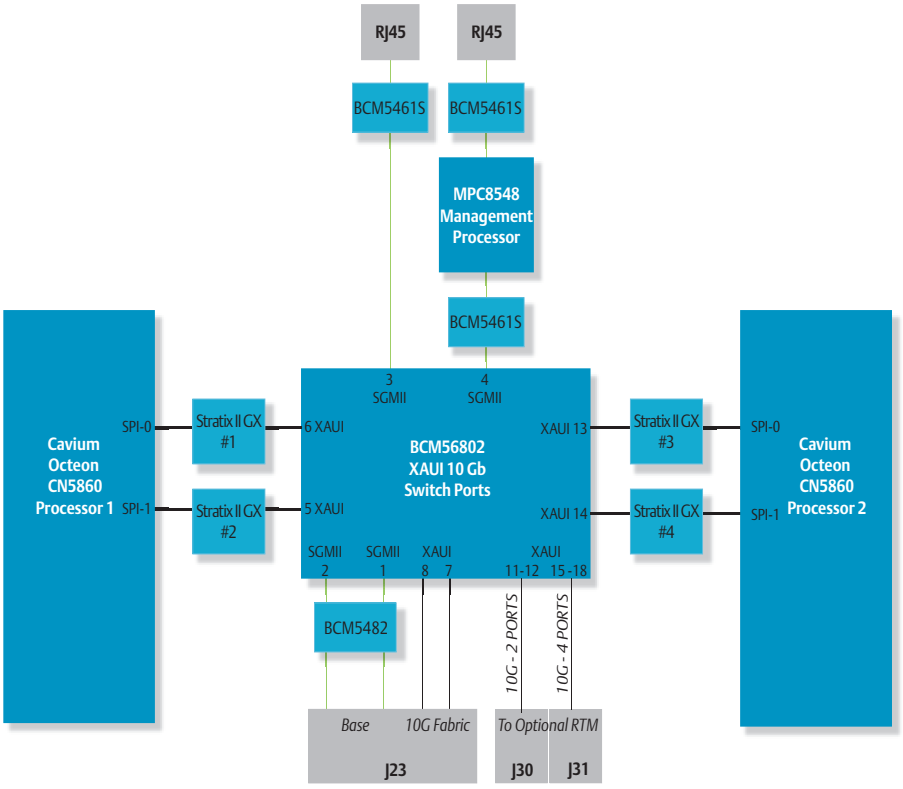
Note: *Proprietary information on the Broadcom switch is not available in this user's manual. Refer to their web site for available documentation.*

ETHERNET SWITCHING

The base interface Ethernet ports are provided by the Broadcom BCM56802 16-port, 10 gigabit (GbE) switch. The SerDes functionality includes 10-Gbps XAUI and 1-Gbps SGMII PHY interfaces. The integrated SerDes complies with the CX-4 standard and PICMG 3.1 standard. The Fabric interface is compliant with PICMG 3.1 Revision 1.0, specifically link option 9 (one 10GBASE-BX4). Switch connectivity consists of the following devices:

- Two 10GbE ports to CN5860 processor complex 1
- Two 10GbE ports to CN5860 processor complex 2
- One GbE port to the front panel (RJ45 connector)
- One GbE port to the MPC8548 management processor complex, then out the front panel (RJ45 connector)
- Two 10 GbE ports to the fabric interface
- Two 1 GbE ports to the base interface
- Two or six 10 GbE ports to the Zone 3 connector (optional RTM)

Figure 6-1: *Ethernet Switching Interface Diagram*



Note: *The phyiscal port numbering starts at 1, as indicated in the figure. However, the software port numbering starts at 0. Therefore, to issue a command to a port, you must subtract 1 from the port numbers shown in the figure.*

Ethernet Transceivers

The BCM5461S is a 10/100/1000BASE-T GbE Ethernet transceiver using the SGMII interface. The BCM5482 consists of two complete 10/100/1000BASE-T GbE transceivers supporting both voice and data simultaneously.

Ethernet Switch Ports

Table 6-1: *Ethernet Switch Ports*

Port:	Interface:	Connection:
1	SGMII 1 GB	PHY to backplane BASE
2	SGMII 1 GB	PHY to backplane BASE

Port:	Interface:	Connection: (continued)
3	SGMII 1 GB	Switch PHY to front panel RJ45 connector
4	SGMII 1 GB	Management processor PHYs to front panel RJ45 connector
5	XAUI 10 GB	Stratix II GX bridge 2
6	XAUI 10 GB	Stratix II GX bridge 1
7	XAUI 10 GB	Back plane Fabric
8	XAUI 10 GB	Back plane Fabric
9	–	not used
10	–	not used
11	XAUI 10 GB	BCM56802 to J30 to optional RTM
12		
13	XAUI 10 GB	Stratix II GX bridge 3
14	XAUI 10 GB	Stratix II GX bridge 4
15	XAUI 10 GB	BCM56802 to J31 to optional RTM
16		
17		
18		

VLAN Setup

The default VLAN configuration is defined in Table 6-2. See page 9-25 for the monitor **vlan** command.

Table 6-2: *VLAN Configuration*

VLAN:	Ports:
1	1, 3, 4
2	6, 7
3	8, 13
4	5, 11
5	12, 14

MPC8548 MANAGEMENT PROCESSOR ETHERNET ADDRESS

The Ethernet address for your board is a unique identifier on a network. The address consists of 48 bits (MAC [47:0]) divided into two equal parts. The upper 24 bits define a unique identifier that has been assigned to Emerson Network Power, Embedded Computing by IEEE. The lower 24 bits are defined by Emerson for identification of each of our products.

The Ethernet address for the ATCA-9305 is a binary number referenced as 12 hexadecimal digits separated into pairs, with each pair representing eight bits. The address assigned to the ATCA-9305 has the following form:

00 80 F9 xx yy zz

00 80 F9 is Emerson's identifier. The last three bytes of the Ethernet address consist of the port (one byte), 0x97(port 1) or 0x98 (port 2), followed by the serial number (two byte hexadecimal). The ATCA-9305 has been assigned the Ethernet address range 00:80:F9:97:00:00 to 00:80:F9:98:FF:FF. The format is shown in Table 6-3.

Table 6-3: *Ethernet Port Address*

Offset:	MAC:	Description:	Ethernet Identifier (hex):
Byte 5	15:0	LSB of (serial number in hex)	–
Byte 4		MSB of (serial number in hex)	–
Byte 3	23:16	Port 1 (TSEC_1) Port 2 (TSEC_2)	0x97 0x98
Byte 2	47:24	Assigned to Emerson by IEEE	0xF9
Byte 1			0x80
Byte 0			0x00

The last two bytes, MAC[15:0], correspond to the following formula: *n* –1000, where *n* is the unique serial number assigned to each board. So if an ATCA-9305 serial number is 1032, the calculated value is 32 (20_{16}), and the default Ethernet port addresses are:

- TSEC_1 MAC address is: 0x00 0x80 0xF9 0x97 0x00 0x20
- TSEC_2 MAC address is: 0x00 0x80 0xF9 0x98 0x00 0x20

Front Panel Ethernet Ports

One MPC8548 PHY (TSEC1) routes to front panel RJ45 connector, P1. The BCM56802 switch PHY (port 3) routes to front panel RJ45 connector, P3. The Ethernet port LEDs (green or yellow) indicate link and activity status, see front panel Fig. 2-1.

Table 6-4: *Front Panel Ethernet Ports*

Pin:	P1 Signal:	P3 Signal:
1	TSEC1_TRD0_P	FP1_TRD0_P
2	TSEC1_TRD0_N	FP1_TRD0_N
3	TSEC1_TRD1_P	FP1_TRD1_P
4	TSEC1_TRD2_P	FP1_TRD2_P
5	TSEC1_TRD2_N	FP1_TRD2_N
6	TSE1C_TRD1_N	FP1_TRD1_N
7	TSEC1_TRD3_P	FP1_TRD3_P
8	TSEC1_TRD3_N	FP1_TRD3_N
9	TSEC1_ACTIVITY (green LED 1)	FP1_ACTIVITY (green LED1)
10	2_5V (yellow LED 1)	2_5V (yellow LED 1)
11	TSEC1_LINKSPD1 (green LED 2)	FP1_LINKSPD1 (green LED 2)

Pin:	P1 Signal:	P3 Signal: (continued)
12	TSEC1_LINKSPD2 (yellow LED 2)	FP1_LINKSPD2 (yellow LED 2)
13	TSEC1_CHSGND	FP1_CHSGND
14	TSEC1_CHSGND	FP1_CHSGND

(blank page)

System Management

The ATCA-9305 provides an intelligent hardware management system, as defined in the *AdvancedTCA Base Specification* (PICMG® 3.0). This system implements an Intelligent Platform Management Controller (IPMC) based on the BMR-H8S-AMCc® reference design from Pigeon Point Systems. It also has an inter-integrated circuit (I^2C) controller to support an Intelligent Platform Management Bus (IPMB) that routes to the ATCA backplane.

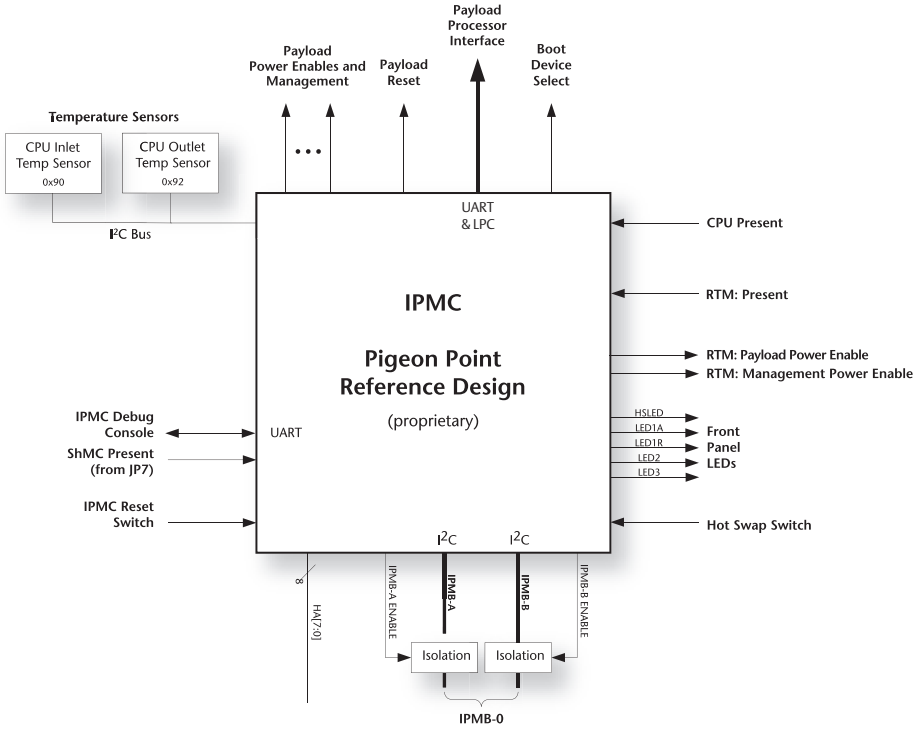
The IPMC implements all the standard Intelligent Platform Management Interface (IPMI) commands and provides hardware interfaces for other system management features such as Hot Swap control, LED control, power negotiation, and temperature and voltage monitoring. The IPMC also supports an EIA-232 interface for serial communications via the Serial Interface Protocol Lite (SIPL) IPMI commands.

IPMC OVERVIEW

The basic features for the IPMC implementation include:

- Conformance with AdvancedTCA Base Specification (PICMG® 3.0)
- Geographical addressing according to PICMG® 3.0
- Ability to read and write Field Replaceable Unit (FRU) data
- Ability to reset IPMC from IPMB
- Ability to read inlet and outlet airflow temperature sensors
- Ability to read payload voltage/current levels
- Ability to send event messages to a specified receiver
- All sensors generate assertion and/or de-assertion event messages
- Support for fault tolerant HPM.1 firmware upgrades
- Support for field updates of firmware via IPMB-0 or the payload interface
- Redundant boot bank capability

Figure 7-1: *IPMC Connections Block Diagram*



IPMI MESSAGING

All IPMI messages contain a Network Function Code field, which defines the category for a particular command. Each category has two codes assigned to it—one for requests and one for responses. The code for a request has the least significant bit of the field set to zero, while the code for a response has the least significant bit of the field set to one. Table 7-1 lists the network function codes (as defined in the IPMI specification) used by the IPMC.

Table 7-1: *Network Function Codes*

Hex Code Value(s):	Name:	Type:	Name:
00, 01	Chassis	chassis device requests/responses	00 = command/request, 01 = response: common chassis control and status functions
02, 03	Bridge	bridge requests/responses	02 = request, 03 = response: message contains data for bridging to the next bus. Typically, the data is another message, which also may be a bridging message. This function is only present on bridge nodes.
04, 05	Sensor/ Event	sensor and event requests/responses	04 = command/request, 05 = response: for configuration and transmission of Event Messages and system Sensors. This function may be present on any node.
06, 07	App	application requests/responses	06 = command/request, 07 = response: message is implementation-specific for a particular device, as defined by the IPMI specification
08, 09	Firmware	firmware transfer requests/responses	08 = command/request, 09 = response: firmware transfer messages match the format of application messages, as determined by the particular device
0A, 0B	Storage	non-volatile storage requests/responses	0A = command/request, 0B = response: may be present on any node that provides nonvolatile storage and retrieval services
0C-2F	reserved	—	reserved: 30 network functions (15 pairs)
30-3F	OEM	—	30 = command/request, 3F = response: vendor specific: 16 network functions (8 pairs). The vendor defines functional semantics for *cmd* and *data* fields. The *cmd* field must hold the same value in requests and responses for a given operation to support IPMI message handling and transport mechanisms. The controller's Manufacturer ID value identifies the vendor or group.

IPMI Completion Codes

All IPMI response messages contain a hexadecimal Completion Code field that indicates the status of the operation.

Table 7-2: *Completion Codes*

Code:	Description:
Generic Completion Codes 00, C0-FF	
00	Command completed normally
C0	Node busy—command could not be processed because command-processing resources are temporarily unavailable
C1	Invalid command—indicates an unrecognized or unsupported command
C2	Command invalid for given LUN
C3	Time-out while processing command, response unavailable
C4	Out of space—command could not be completed because of a lack of storage space required to execute the given command operation
C5	Reservation canceled or invalid Reservation ID
C6	Request data truncated
C7	Request data length invalid
C8	Request data field length limit exceeded
C9	Parameter out of range—one or more parameters in the data field of the Request are out of range. This is different from *Invalid data field* code (CC) because it indicates that the erroneous field(s) has a contiguous range of possible values.
CA	Cannot return number of requested data bytes
CB	Requested sensor, data, or record not present
CC	Invalid data field in Request
CD	Command illegal for specified sensor or record type
CE	Command response could not be provided
CF	Cannot execute duplicated request—for devices that cannot return the response returned for the original instance of the request. These devices should provide separate commands that allow the completion status of the original request to be determined. An Event Receiver does not use this completion code, but returns the 00 completion code in the response to (valid) duplicated requests.
D0	Command response could not be provided, SDR Repository in update mode
D1	Command response could not be provided, device in firmware update mode
D2	Command response could not be provided, Baseboard Management Controller (BMC) initialization or initialization agent in progress
D3	Destination unavailable—cannot deliver request to selected destination. (This code can be returned if a request message is targeted to SMS, but receive message queue reception is disabled for the particular channel.)
D4	Cannot execute command, insufficient privilege level
D5	Cannot execute command, parameter(s) not supported in present state
FF	Unspecified error

Code:	Description: (continued)
Device-Specific (OEM) Codes 01-7E	
01-7E	Device specific (OEM) completion codes—command-specific codes (also specific for a particular device and version). Interpretation of these codes requires prior knowledge of the device command set.
Command-Specific Codes 80-BE	
80-BE	Standard command-specific codes—reserved for command-specific completion codes (described in this chapter)

IPMB PROTOCOL

The IPMB message protocol is designed to be robust and support many different physical interfaces. The IPMC supports messages over the IPMB interface. Messages are defined as either a request or a response, as indicated by the least significant bit in the Network Function Code of the message.

Table 7-3: *Format for IPMI Request Message*

Byte:	Bits:							
	7	6	5	4	3	2	1	0
1	rsSA							
2	Network Function (netFn)						rsLUN	
3	Checksum							
4	rqSA							
5	rqSeq						rqLUN	
6	Command							
7:N	Data							
N+1	Checksum							

- The first byte contains the responder's Slave Address, **rsSA**.
- The second byte contains the Network Function Code, **netFn**, and the responder's Logical Unit Number, **rsLUN**.
- The third byte contains the two's-complement checksum for the first two bytes.
- The fourth byte contains the requester's Slave Address, **rqSA**.
- The fifth byte contains the requester's Sequence Number, **rqSeq**, and requester's Logical Unit Number, **rqLUN**. The Sequence number may be used to associate a specific response to a specific request.
- The sixth byte contains the Command Number.

- The seventh byte and beyond contain parameters for specific commands (if required).
- The final byte is the two's-complement checksum of all of the message data after the first checksum.

An IPMI response message (see Table 7-4) is similar to an IPMI request message. The main difference is that the seventh byte contains the Completion Code, and the eighth byte and beyond hold data received from the controller (rather than data to send to the controller). Also, the Slave Address and Logical Unit Number for the requester and responder are swapped.

Table 7-4: *Format for IPMI Response Message*

Byte:	Bits:							
	7	6	5	4	3	2	1	0
1	rqSA							
2	Network Function (netFn)						rqLUN	
3	Checksum							
4	rsSA							
5	rsSeq						rsLUN	
6	Command							
7	Completion Code							
8:N	Data							
N+1	Checksum							

SIPL PROTOCOL

The IPMC supports the Serial Interface Protocol Lite (SIPL) protocol. It supports raw IPMI messages in SIPL and handles these messages the same way as it handles IPMI messages from the IPMB-0 bus, except that the replies route to either the payload or serial debug interface. Messages are entered as case-insensitive hex-ASCII pairs, separated optionally by a space, as shown in the following examples:

```
[18 00 22]<newline>

[180022]<newline>
```

The IPMC does not, however, support SIPL ASCII text commands, as defined by the IPMI specification.

The IPMC does support Pigeon Point Systems extension commands, implemented as OEM IPMI commands. These commands use Network Function Codes 2E/2F (hex), and the message body is transferred similarly to raw IPMI messages, as described previously.

The following figures show an example of an extension command request and response, respectively.

Figure 7-2: *Extension Command Request Example*

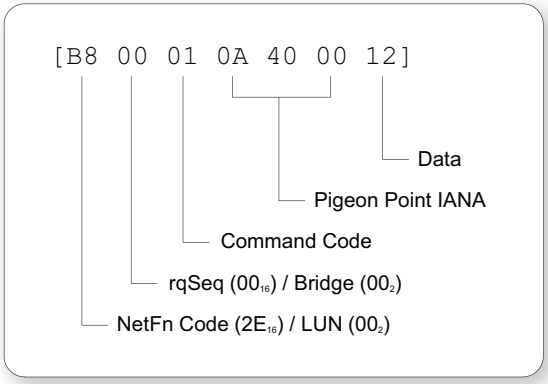
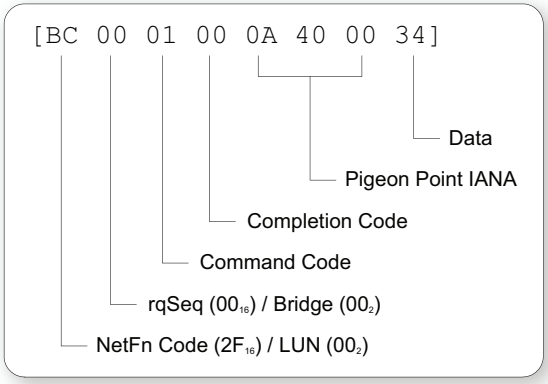


Figure 7-3: *Extension Command Response Example*



MESSAGE BRIDGING

The Message Bridging facility is responsible for bridging messages between various interfaces of the ATCA-9305 IPMI. The message bridging is implemented via the standard **Send Message** command.

The ATCA-9305 IPMC also supports message bridging between the Payload Interface and IPMB-0, which allows the payload to send custom messages to and receive them from other shelf entities, such as the shelf manager. Message bridging is implemented using the **Send/Get Message** commands and also via LUN 10 of the ATCA-9305 IPMC.

The following example illustrates how the **Send/Get Message** and **Get Address Info** commands can be used by the payload software to get the physical location of the board in the shelf:

1 The payload software sends the **Get Address Info** command to the BMR-H8S-AMCc, requesting address information for FRU device 0. Using the SIPL protocol:

```
[B0 xx 01 00]
```

2 The BMR-H8S-AMCc returns its IPMB address in the Get Address Info reply. In this example, 72_{16} is the IPMB-0 address of the IPMC.

```
{B4 00 01 00 00 FF 72 FF 00 01 07]
```

3 The payload software composes a **Get Address Info** command requesting the responder to provide its addressing information for FRU device 0. The request is composed in the IPMB format. The responder address is set to 20_{16} (for the shelf manager). The requester address is set to the value obtained in the previous step.

```
{20 B0 30 72 00 01 00 8D]
```

4 The payload software forwards the command composed in the previous step to the shelf manager using the **Send Message** command. The Send/Get Message in SIPL format is:

```
[18 xx 34 40 20 B0 30 72 00 01 00 8D]
```

5 The BMR-H8S-AMCc firmware sends the Get Address Info request to the shelf manager, waits for a reply to this request, and sends this reply to the payload software in the Send/Get Message response.

```
[1C 00 34 00 72 B4 DA 20 00 01 00 00 41 82 FF 00 FF 00 1E]
```

6 The payload software extracts the Get Address info reply from the Send/Get Message response and retrieves the physical address of the board from it.

The second message bridging implementation, bridging via LUN 10, allows the payload to receive responses to requests sent to IPMB-0 via the **Send Message** command with request tracking disabled, as well as receive requests from IPMB-0. To provide this functionality, the ATCA-9305 IPMC places all messages coming to LUN 10 from IPMB-0 in a dedicated Receive Message Queue, and those messages are processed by the payload instead of the IPMC firmware. To read messages from the Receive Message Queue, the payload software uses the standard **Get Message** command. The payload software is notified about messages coming to LUN 10 via the **Get Status** command of the SIPL protocol and the payload notification mechanism, or, if the LPC/KCS-based Payload Interface is used, using the KCS interrupt. The Receive Message Queue of the ATCA-9305 IPMC is limited to 128 bytes, which is sufficient for storing at least three IPMB messages, but may be not enough for a larger number of messages. Taking this into account, the payload software must read messages from the queue as fast as possible, caching them on the on-carrier payload side for further han-

dling, if it is necessary. If the Receive Message Queue is full, the ATCA-9305 IPMC rejects all requests coming to LUN 10 with the C0h (Node Busy) completion code and discards all responses coming to this LUN.

STANDARD COMMANDS

The Intelligent Peripheral Management Controller (IPMC) supports standard IPMI commands to query board information and to control the behavior of the board. These commands provide a means to:

- identify the controller
- reset the controller
- return the controller's self-test results
- read and write the controller's SROMs
- read the temperature, voltage, and watchdog sensors
- get specific information, such as thresholds, for each sensor
- read and write the Field Replaceable Unit (FRU) data
- reserve and read the Sensor Data Record (SDR) repository
- configure event broadcasts
- bridge an IPMI request to the public IPMB and return the response

Table 7-5 lists the IPMI commands supported by the IPMC along with the hexadecimal values for each command's Network Function Code (**netFn**), Logical Unit Number (**LUN**), and Command Code (**Cmd**):

Table 7-5: *IPMC IPMI Commands*

Command:	netFn:	LUN:	Cmd:
Set System Boot Options	Chassis	01, 01	07
Get System Boot Options	Chassis	01, 01	08
Set Event Receiver	Sensor/Event	04, 05	00
Get Event Receiver	Sensor/Event	04, 05	01
Platform Event (Event Message)	Sensor/Event	04, 05	02
Get Device SDR Information	Sensor/Event	04, 05	20
Get Device SDR	Sensor/Event	04, 05	21
Reserve Device SDR Repository	Sensor/Event	04, 05	22
Get Sensor Reading Factors	Sensor/Event	04, 05	23
Set Sensor Hysteresis	Sensor/Event	04, 05	24
Get Sensor Hysteresis	Sensor/Event	04, 05	25

Command: (continued)	netFn:	LUN:	Cmd:
Set Sensor Thresholds	Sensor/Event	04, 05	26
Get Sensor Thresholds	Sensor/Event	04, 05	27
Set Sensor Event Enable	Sensor/Event	04, 05	28
Get Sensor Event Enable	Sensor/Event	04, 05	29
Rearm Sensor Events	Sensor/Event	04, 05	2A
Get Sensor Event Status	Sensor/Event	04, 05	2B
Get Sensor Reading	Sensor/Event	04, 05	2D
Set Sensor Type	Sensor/Event	04, 05	2E
Get Sensor Type	Sensor/Event	04, 05	2F
Get Device ID	Application	06, 07	01
Broadcast 'Get Device ID'	Application	06, 07	01
Cold Reset	Application	06, 07	02
Warm Reset	Application	06, 07	03
Get Self Test Results	Application	06, 07	04
Get Device GUID	Application	06, 07	08
Reset Watchdog Timer	Application	06, 07	22
Set Watchdog Timer	Application	06, 07	24
Get Watchdog Timer	Application	06, 07	25
Send Message	Application	06, 07	34
Get FRU Inventory Area Info	Storage	0A, 0B	10
Read FRU Data	Storage	0A, 0B	11
Write FRU Data	Storage	0A, 0B	12
Get PICMG Properties	PICMG	2C, 2D	00
Get Address Info	PICMG	2C, 2D	01
FRU Control	PICMG	2C, 2D	04
Get FRU LED Properties	PICMG	2C, 2D	05
Get LED Color Capabilities	PICMG	2C, 2D	06
Set FRU LED State	PICMG	2C, 2D	07
Get FRU LED State	PICMG	2C, 2D	08
Set IPMB State	PICMG	2C, 2D	09
Set FRU Activation Policy	PICMG	2C, 2D	0A
Get FRU Activation Policy	PICMG	2C, 2D	0B
Set FRU Activation	PICMG	2C, 2D	0C
Get Device Locator Record ID	PICMG	2C, 2D	0D
Set Port State	PICMG	2C, 2D	0E
Get Port State	PICMG	2C, 2D	0F
Compute Power Properties	PICMG	2C, 2D	10
Set Power Level	PICMG	2C, 2D	11

Command: (continued)	netFn:	LUN:	Cmd:
Get Power Level	PICMG	2C, 2D	12
Bused Resource (Release, Query, Force, Bus Free)	PICMG	2C, 2D	17

The IPMC implements many standard IPMI commands. For example, software can use the watchdog timer commands to monitor the system's health. Normally, the software resets the watchdog timer periodically to prevent it from expiring. The IPMI specification allows for different actions such as reset, power off, and power cycle, to occur if the timer expires. The watchdog's 'timer use' fields can keep track of which software (Operating System, System Management, etc.) started the timer. Also, the time-out action and 'timer use' information can be logged automatically to the System Event Log (SEL) when the time-out occurs. Refer to the IPMI specification (listed in Table 1-2) for details about each command's request and response data. The IPMC also implements ATCA commands, see the ATCA Base Specification (PICMG 3.0).

OEM BOOT OPTIONS

The **Set System Boot Options** and **Get System Boot Options** commands provide a means to set/retrieve the boot options. The IPMI specification defines a set of standard boot option parameters. In addition, the specification includes a range of numbers (96-127) for OEM extensions. Emerson utilizes this area for OEM function extensions, such as boot bank selection and POST configuration. The following table describes these extensions:

Table 7-6: *Emerson Boot Option Parameters*

Parameter:	#	Parameter Data:
Boot Bank (non-volatile)	96	data 1 – Set Selector. This is the processor ID for which the boot option is to be set. data 2 – Boot Bank Selector. This parameter is used to indicate the boot bank from which the payload will boot. 00h = Primary (i.e., default) Boot Bank is selected. 01h = Secondary Boot Bank is selected. 02h-FFh = unused
POST Type (non-volatile)	97	data 1 – Set Selector. This is the processor ID for which the boot option is to be set. data 2 – PSOT Type Selector. This parameter is used to specify the POST type that the payload boot firmware will execute. 00h = Short POST 01h = Long POST 02h-FFh = unused

IPMC WATCHDOG TIMER COMMANDS

The IPMC implements a standardized 'Watchdog Timer' that can be used for a number of system time-out functions by System Management Software (SMS) or by the monitor. Setting a time-out value of zero allows the selected time-out action to occur immediately. This provides a standardized means for devices on the IPMB to perform emergency recovery actions.

Table 7-7: *IPMC Watchdog Timer Commands*

Command:	See Page:	Optional/Mandatory:
Reset Watchdog Timer	7-14	M
Set Watchdog Timer	7-14	M
Get Watchdog Timer	7-16	M

Watchdog Timer Actions

The following actions are available on expiration of the Watchdog Timer:

- System Reset
- System Power Off

The System Reset and System Power Off on time-out selections are mutually exclusive. The watchdog timer is stopped whenever the system is powered down. A command must be sent to start the timer after the system powers up.

Watchdog Timer Use Field and Expiration Flags

The watchdog timer provides a 'timer use' field that indicates the current use assigned to the watchdog timer. The watchdog timer provides a corresponding set of 'timer use expiration' flags that are used to track the type of time-out(s) that had occurred.

The time-out use expiration flags retain their state across system resets and power cycles, as long as the IPMC remains powered. The flags are normally cleared solely by the **Set Watchdog Timer** command; with the exception of the "don't log" flag, which is cleared after every system hard reset or timer time-out.

The Timer Use fields indicate:

Monitor FRB-2 Time-out:

A Fault-resilient Booting, level 2 (FRB-2) time-out has occurred. This indicates that the last system reset or power cycle was due to the system time-out during POST, presumed to be caused by a failure or hang related to the bootstrap processor.

Monitor POST Time-out:

In this mode, the time-out occurred while the watchdog timer was being used by the monitor for some purpose other than FRB-2 or OS Load Watchdog.

OS Load Time-out: The last reset or power cycle was caused by the timer being used to 'watchdog' the interval from 'boot' to OS up and running. This mode requires system management software, or OS support. The monitor should clear this flag if it starts this timer during POST.

SMS 'OS Watchdog' Time-out:

This indicates that the timer was being used by System Management Software (SMS). During run-time, SMS starts the timer, then periodically resets it to keep it from expiring. This periodic action serves as a 'heartbeat' that indicates that the OS (or at least the SMS task) is still functioning. If SMS hangs, the timer expires and the IPMC generates a system reset. When SMS enables the timer, it should make sure the 'SMS' bit is set to indicate that the timer is being used in its 'OS Watchdog' role.

OEM: This indicates that the timer was being used for an OEM-specific function.

Using the Timer Use Field and Expiration Flags

The software that sets the Timer Use field is responsible for managing the associated Timer Use Expiration flag. For example, if System Management Software (SMS) sets the timer use to "SMS/OS Watchdog," then that same SMS is responsible for acting on and clearing the associated Timer Use Expiration flag.

In addition, software should *only* interpret or manage the expiration flags for watchdog timer uses that it set. For example, the monitor should not report watchdog timer expirations or clear the expiration flags for non-monitor uses of the timer. This is to allow the software that did set the Timer Use to see that a matching expiration occurred.

Watchdog Timer Event Logging

By default, the IPMC will automatically log the corresponding sensor-specific watchdog sensor event when a timer expiration occurs. A "don't log" bit is provided to temporarily disable the automatic logging. The "don't log" bit is automatically cleared (logging re-enabled) whenever a timer expiration occurs.

Monitor Support for Watchdog Timer

If a system "Warm Reset" occurs, the watchdog timer may still be running while the monitor executes POST. Therefore, the monitor should take steps to stop or restart the watchdog timer early in POST. Otherwise, the timer may expire later during POST or after the OS has booted.

Reset Watchdog Timer Command

The **Reset Watchdog Timer** command is used for starting and restarting the Watchdog Timer from the initial countdown value that was specified in the **Set Watchdog Timer** command.

If a pretime-out interrupt has been configured, the **Reset Watchdog Timer** command will not restart the timer once the pretime-out interval has been reached. The only way to stop the timer once it has reached this point is via the **Set Watchdog Timer** command.

Table 7-8: *Reset Watchdog Timer Command*

Type:	Byte:	Data Field:
Request Data	–	–
Response Data	1	Completion Code

Set Watchdog Timer Command

The **Set Watchdog Timer** command is used for initializing and configuring the watchdog timer. The command is also used for stopping the timer.

If the timer is already running, the **Set Watchdog Timer** command stops the timer (unless the "don't stop" bit is set) and clears the Watchdog pretime-out interrupt flag (see **Get Message Flags** command in the IPMI specification v1.5). IPMC hard resets, system hard resets, and the **Cold Reset** command also stop the timer and clear the flag.

Byte 1: This selects the timer use and configures whether an event will be logged on expiration.

Byte 2: This selects the time-out action and pretime-out interrupt type.

Byte 3: This sets the pretime-out interval. If the interval is set to zero, the pretime-out action occurs concurrently with the time-out action.

Byte 4: This clears the Timer Use Expiration flags. A bit set in byte 4 of this command clears the corresponding bit in byte 5 of the **Get Watchdog Timer** command.

Bytes 5 and 6: These hold the least significant and most significant bytes, respectfully, of the countdown value. The Watchdog Timer decrement is one count/100 ms. The counter expires when the count reaches zero. If the counter is loaded with zero and the **Reset Watchdog** command is issued to start the timer, the associated timer events occur immediately.

Table 7-9: *Set Watchdog Timer Command*

Type:	Byte:	Data Field:
Request Data	1	Timer Use [7] 1b=don't log [6] 1b=the don't stop timer on **Set Watchdog Timer** command (new for IPMI v1.5) new parameters take effect immediately. If timer is already running, countdown value will get set to given value and countdown will continue from that point. If timer is already stopped, it will remain stopped. If the pretime-out interrupt bit is set, it will get cleared.[1] 0b=timer stops automatically when **Set Watchdog Timer** command is received [5:3] reserved [2:0] timer use (logged on expiration when "don't log" bit = 0b) 000b=reserved 001b=Monitor FRB-2 010b=Monitor/POST 011b=OS Load 100b=SMS/OS 101b=OEM 110b-111b=reserved
	2	Timer Actions [7] reserved [6:4] pretime-out interrupt (logged on expiration when "don't log" bit = 0b) 000b=none 001b=SMI 010b=NMI/Diagnostic Interrupt 011b=Messaging Interrupt (this is the same interrupt as allocated to the messaging interface) 100b, 111b =reserved [3] reserved [2:0] time-out action 000b=no action 001b=Hard Reset 010b=Power Down 011b=Power Cycle 100b, 111b=reserved
	3	Pretime-out interval in seconds, '1' based

Type:	Byte:	Data Field: (continued)
Request Data (continued)	4	Timer Use Expiration flags clear (0b=leave alone, 1b=clear timer use expiration bit) [7] reserved [6] reserved [5] OEM [4] SMS/OS [3] OS Load [2] Monitor/POST [1] Monitor FRB-2 [0] reserved
	5	Initial countdown value, lsbyte (100 ms/count)
	6	Initial countdown value, msbyte
Response Data	1	Completion Code

1. Potential race conditions exist with implementation of this option. If the Set Watchdog Timer command is sent just before a pretime-out interrupt or time-out is set to occur, the time-out could occur before the command is executed. To avoid this condition, it is recommended that software set this value no closer than three counts before the pretime-out or time-out value is reached.

Get Watchdog Timer Command

This command retrieves the current settings and present countdown of the watchdog timer. The Timer Use Expiration flags in byte 5 retain their states across system resets and system power cycles. With the exception of bit 6 in the Timer Use byte, the Timer Use Expiration flags are cleared using the **Set Watchdog Timer** command. They may also become cleared because of a loss of IPMC power, firmware update, or other cause of IPMC hard reset. Bit 6 of the Timer Use byte is automatically cleared to 0b whenever the timer times out, is stopped when the system is powered down, enters a sleep state, or is reset.

Table 7-10: *Get Watchdog Timer Command*

Type:	Byte:	Data Field:
Request Data	–	–
Response Data	1	Completion Code

Type:	Byte:	Data Field: (continued)
Response Data	2	Timer Use [7] 1b=don't log [6] 1b=timer is started (running) 0b=timer is stopped [5:3] reserved [2:0] timer use (logged on expiration if "don't log" bit = 0) 000b=reserved 001b=Monitor FRB-2 010b=Monitor/POST 011b=OS Load 100b=SMS/OS 101b=OEM 110b, 111b=reserved
	3	Timer Actions [7] reserved [6:4] pretime-out interrupt 000b=none 001b=SMI 010b=NMI/Diagnostic Interrupt 011b=Messaging Interrupt (this would be the same interrupt as allocated to the messaging interface) 100b, 111b =reserved [3] reserved [2:0] time-out action 000b=no action 001b=Hard Reset 010b=Power Down 011b=Power Cycle 100b, 111b=reserved
	4	Pretime-out interval in seconds, '1'based
	5	Timer Use Expiration flags (1b=timer expired while associated 'use' was selected) [7] reserved [6] reserved [5] OEM [4] SMS/OS [3] OS Load [2] Monitor/POST [1] Monitor FRB-2 [0] reserved
	6	Initial countdown value, lsbyte (100 ms/count)
	7	Initial countdown, msbyte

Type:	Byte:	Data Field: (continued)
Response Data	8	Present countdown value, lsbyte. The initial countdown value and present countdown values should match immediately after the countdown is initialized via a **Set Watchdog Timer** command and after a **Reset Watchdog Timer** has been executed. Note that internal delays in the IPMC may require software to delay up to 100 ms before seeing the countdown value change and be reflected in the **Get Watchdog Timer** command.
	9	Present countdown value, msbyte

FRU LEDS

This section describes the front panel LEDs controlled by the IPMC and documents how to control each LED with the standard FRU LED commands. Reference the *PICMG® 3.0 Revision 2.0 AdvancedTCA® Base Specification* for more detailed information.

The ATCA-9305 has four Light-Emitting Diodes (LEDs) on the front panel. See Fig. 2-1 for their location.

Table 7-11: *FRU LEDs*

LEDs:	ID (hex):	Reference Designator:	Description:
Hot Swap	00	CR57	The blue Hot Swap LED displays four states: On–the board can be safely extracted Off–the board is operating and not safe for extraction, Long blink–insertion is in progress Short blink–requesting permission for extraction
OOS	01	CR54	The Out Of Service programmable LED controlled by the IPMI controller is either red (North America) or amber (Europe). When lit, this LED indicates the ATCA-9305 is in a failed state.
2	02	CR55	The green LED is user defined, but frequently is used as an In Service indicator. When used as an In Service indicator, a lit LED indicates that the ATCA-9305 is functioning properly.
3	03	CR56	The amber LED is user defined.

Get FRU LED Properties Command

This command allows software to determine which LEDs are under IPMC control.

Table 7-12: *Get FRU LED Properties Command*

Type:	Byte:	Data Field:
Request Data	1	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.
	2	FRU Device ID
Response Data	1	Completion Code
	2	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.
	3	General Status LED Properties–indicates the FRU's ability to control the four general status LEDs. When a bit is set, the FRU can control the associated LED. Bits [7:4] reserved, set to 0 Bit [3] LED3 Bit [2] LED2 Bit [1] LED1 Bit [0] Blue LED
	4	Application Specific LED Count–is the number of application specific LEDs under IPMC control. 00h-FBh Number of application-specific LEDs under IPMC control. If none are present, this field is 00h. FCh-FFh reserved

Get LED Color Capabilities Command

LED 1 can be either red or amber, this command is used to determine the valid color prior to issuing a **Set FRU LED State** command.

Table 7-13: *Get LED Color Capabilities Command*

Type:	Byte:	Data Field:
Request Data	1	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.
	2	FRU Device ID
	3	LED ID FFh reserved

Type:	Byte:	Data Field: (continued)
Response Data	1	Completion Code CCh If the LED ID contained in the Request data is not present on the FRU
	2	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.
	3	LED Color Capabilities–when a bit is set, the LED supports the color. Bit [7] reserved, set to 0 Bit [6] LED supports white Bit [5] LED supports orange Bit [4] LED supports amber Bit [3] LED supports green Bit [2] LED supports red Bit [1] LED supports blue Bit [0] reserved, set to 0
	4	Default LED Color in Local Control State Bit [7] reserved, set to 0 Bits [3:0] 0h reserved 1h Blue 2h Red 3h Green 4h Amber 5h Orange 6h White 7h-Fh reserved
	5	Default LED Color in Override State Bit [7] reserved, set to 0 Bits [3:0] 0h reserved 1h Blue 2h Red 3h Green 4h Amber 5h Orange 6h White 7h-Fh reserved

Set FRU LED State Command

The **Set FRU LED State** command allows the state of the FRU LEDs to be controlled by the management system.

Table 7-14: *Set FRU LED State Command*

Type:	Byte:	Data Field:
Request Data	1	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.
	2	FRU Device ID
	3	LED ID 00h Blue LED (Hot Swap) 01h LED 1 (OOS) 02h LED 2 03h LED 3 04h-FEh OEM defined LEDs FFh Lamp Test (all LEDs under management control are addressed)
	4	LED Function 00h LED off override 01h-FAh LED blinking override FBh Lamp Test state Turn on LED specified in byte 3 for the duration specified in byte 5, then return to the highest priority state. FCh LED state restored to Local Control state FDh-FEh reserved FFh LED on override
	5	On Duration LED on-time is measured in tens of milliseconds Lamp Test time in hundreds of milliseconds if byte 4=FBh, time value must be less than 128. Other values when Byte 4=FBh are reserved. Otherwise, this field is ignored and shall be set to 0h.

Type:	Byte:	Data Field: (continued)
Request Data	6	Color When Illuminated–sets the override color when *LED Function* is 01h-FAh and FFh. This byte sets the Local Control color when LED Function is FCh. This byte may be ignored during Lamp Test or may be used to control the color during the lamp test when *LED Function* is FBh. Bits [7:4] reserved, set to 0 Bits [3:0] 0h reserved 1h Use Blue 2h Use Red 3h Use Green 4h Use Amber 5h Use Orange 6h Use White 7h-Dh reserved Eh Do not change Fh Use default color
Response Data	1	Completion Code
	2	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.

Get FRU LED State Command

The **Get FRU LED State** command allows the state of the FRU LEDs to be controlled by the management system.

Table 7-15: *Get FRU LED State Command*

Type:	Byte:	Data Field:
Request Data	1	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.
	2	FRU Device ID
	3	LED ID 00h Blue LED (Hot Swap) 01h LED 1 (OOS) 02h LED 2 03h LED 3 04h-FEh OEM defined LEDs FFh reserved
Response Data	1	Completion Code
	2	PICMG Identifier–indicates that this is a PICMG defined group extension command. Use value 00h.

Type:	Byte:	Data Field: (continued)
Response Data	3	LED States Bits [7:3] reserved, set to 0 Bit [2] 1b if Lamp Test has been enabled Bit [1] 1b if override state has been enabled Bit [2] 1b if IPMC has a Local control state
	4	Local Control LED Function 00h LED is off (default if Local Control not supported) 01h-FAh LED is blinking Off duration specified by this byte, on duration specified by byte 5 (in tens of milliseconds) FBh-FEh reserved FFh LED is on
	5	On Duration LED on-time is measured in tens of milliseconds Lamp Test time in hundreds of milliseconds if byte 4=FBh, time value must be less than 128. Other values when Byte 4=FBh are reserved. Otherwise, this field is ignored and shall be set to 0h.
	6	Local Control Color Bits [7:4] reserved, set to 0 Bits [3:0] 0h reserved 1h Blue 2h Red 3h Green 4h Amber 5h Orange 6h White 7h-Fh reserved
	7	Override State LED Function–is required if either override state or Lamp Test is in effect. 00h LED override state is off 01h-FAh LED override state is blinking Off duration is specified by this byte, on duration specified by byte 8 (in tens of milliseconds) FBh-FEh reserved FFh LED override state is on
	8	Override State On Duration–is required if either override state or Lamp Test is in effect (in tens of milliseconds).

Type:	Byte:	Data Field: (continued)
Response Data	9	Override State Color Bits [7:4] reserved, set to 0 Bits [3:0] 0h reserved 1h Blue 2h Red 3h Green 4h Amber 5h Orange 6h White 7h-Fh reserved
	10	Lamp Test Duration–is optional if Lamp Test is not in effect (hundreds of milliseconds).

VENDOR COMMANDS

The IPMC supports additional IPMI commands that are specific to Pigeon Point and/or Emerson. This section provides detailed descriptions of those extension or SIPL commands.

Table 7-16: *Vendor Command Summary*

Command:	netFn:	LUN:	Cmd:
Get Status	OEM	2E, 2F	00
Get Serial Interface Properties	OEM	2E, 2F	01
Set Serial Interface Properties	OEM	2E, 2F	02
Get Debug Level	OEM	2E, 2F	03
Set Debug Level	OEM	2E, 2F	04
Get Hardware Address	OEM	2E, 2F	05
Set Hardware Address	OEM	2E, 2F	06
Get Handle Switch	OEM	2E, 2F	07
Set Handle Switch	OEM	2E, 2F	08
Get Payload Communication Time-Out	OEM	2E, 2F	09
Set Payload Communication Time-Out	OEM	2E, 2F	0A
Enable Payload Control	OEM	2E, 2F	0B
Disable Payload Control	OEM	2E, 2F	0C
Reset IPMC	OEM	2E, 2F	0D
Hang IPMC	OEM	2E, 2F	0E
Bused Resource Control	OEM	2E, 2F	0F
Bused Resource Status	OEM	2E, 2F	10
Graceful Reset	OEM	2E, 2F	11
Diagnostic Interrupt Results	OEM	2E, 2F	12
Get Payload Shutdown Time-Out	OEM	2E, 2F	15
Set Payload Shutdown Time-Out	OEM	2E, 2F	16

Command: (continued)	netFn:	LUN:	Cmd:
Set Local FRU LED State	OEM	2E, 2F	18
Get Local FRU LED State	OEM	2E, 2F	19
Update Discrete Sensor	OEM	2E, 2F	1A
Update Threshold Sensor	OEM	2E, 2F	1B
Reserved for Message Listeners	OEM	30, 31	10
Add Message Listener	OEM	30, 31	11
Remove Message Listener	OEM	30, 31	12
Get Message Listener List	OEM	30, 31	13
Update Firmware Progress Sensor	OEM	30, 31	F0

Get Status

The IPMC firmware notifies the payload about changes of all status bits except for bits 0-2 by sending an unprintable character (ASCII 07, BELL) over the Payload Interface. The payload is expected to use the **Get Status** command to identify pending events and other SIPL commands to provide a response (if necessary). The event notification character is sent in a synchronous manner, and does not appear in the contents of SIPL messages sent to the payload.

Table 7-17: *Get Status Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Type:	Byte:	Data Field: (continued)
Response Data	5	Bit [7] Graceful Reboot Request If set to 1, indicates that the payload is requested to initiate the graceful reboot sequence Bit [6] Diagnostic Interrupt Request If set to 1, indicates that a payload diagnostic interrupt request has arrived Bit [5] Shutdown Alert If set to 1, indicates that the payload is going to be shutdown Bit [4] Reset Alert If set to 1, indicates that the payload is going to be reset Bit [3] Sensor Alert If set to 1, indicates that at least one of the IPMC sensors detects threshold crossing Bits [2:1] Mode The current IPMC modes are defined as: 0 Normal 1 Standalone 2 Manual Standalone Bit [0] Control If set to 0, the IPMC control over the payload is disabled
	6	Bits [4:7] Metallic Bus 2 Events These bits indicate pending Metallic Bus 2 requests arrived from the carrier controller: 0 Metallic Bus 2 Query 1 Metallic Bus 2 Release 2 Metallic Bus 2 Force 3 Metallic Bus 2 Free Bits [0:3] Metallic Bus 1 Events These bits indicate pending Metallic Bus 1 requests arrived from the carrier controller: 0 Metallic Bus 1 Query 1 Metallic Bus 1 Release 2 Metallic Bus 1 Force 3 Metallic Bus 1 Free

Type:	Byte:	Data Field: (continued)
Response Data	7	Bits [4:7] Clock Bus 2 Events These bits indicate pending Clock Bus 2 requests arrived from the carrier controller: 0 Clock Bus 2 Query 1 Clock Bus 2 Release 2 Clock Bus 2 Force 3 Clock Bus 2 Free Bits [0:3] Clock Bus 1 Events These bits indicate pending Clock Bus 1 requests arrived from the carrier controller: 0 Clock Bus 1 Query 1 Clock Bus 1 Release 2 Clock Bus 1 Force 3 Clock Bus 1 Free
	8	Bits [4:7] reserved Bits [0:3] Clock Bus 3 Events These bits indicate pending Clock Bus 3 requests arrived from the carrier controller: 0 Clock Bus 3 Query 1 Clock Bus 3 Release 2 Clock Bus 3 Force 3 Clock Bus 3 Free

Get Serial Interface Properties

The **Get Serial Interface Properties** command is used to get the properties of a particular serial interface.

Table 7-18: *Get Serial Interface Properties Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Interface ID 0 Serial Debug Interface 1 Payload Interface
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Type:	Byte:	Data Field: (continued)
Response Data	5	Bit [7] Echo On If this bit is set, the IPMC enables echo for the given serial interface Bits [6:4] reserved Bits [3:0] Baud Rate ID The baud rate ID defines the interface baud rate as follows: 0 9600 bps 1 19200 bps 2 38400 bps 3 57600 bps (unsupported) 4 115200 bps (unsupported)

Set Serial Interface Properties

The **Set Serial Interface Properties** command is used to set the properties of a particular serial interface.

Table 7-19: *Set Serial Interface Properties Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Interface ID 0 Serial Debug Interface 1 Payload Interface
	5	Bit [7] Echo On If this bit is set, the IPMC enables echo for the given serial interface Bits [6:4] reserved Bits [3:0] Baud Rate ID The baud rate ID defines the interface baud rate as follows: 0 9600 bps 1 19200 bps 2 38400 bps 3 57600 bps (unsupported) 4 115200 bps (unsupported)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Get Debug Level

The **Get Debug Level** command gets the current debug level of the IPMC firmware.

Table 7-20: *Get Debug Level Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	5	Bits [7:5] reserved Bit [4] IPMB Dump Enable If set to 1, the IPMC provides a trace of IPMB messages that are arriving to/going from the IPMC via IPMB-0 or IPMB -L Bit [3] Payload Logging Enable If set to 1, the IPMC provides a trace of SIPL activity on the Payload interface onto the Serial Debug interface Bit [2] Alert Logging Enable If set to 1, the IPMC outputs important alert messages onto the Serial Debug interface Bit [1] Low-level Error Logging Enable If set to 1, the IPMC outputs low-level error/diagnostic messages onto the Serial Debug interface Bit [0] Error Logging Enable If set to 1, the IPMC outputs error/diagnostic messages onto the Serial Debug interface

Set Debug Level

The **Set Debug Level** command sets the current debug level of the IPMC firmware.

Table 7-21: *Set Debug Level Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Type:	Byte:	Data Field: (continued)
Request Data	4	Bits [7:5] reserved Bit [4] IPMB Dump Enable If set to 1, the IPMC provides a trace of IPMB messages that are arriving to/going from the IPMC via IPMB-0 or IPMB-L Bit [3] Payload Logging Enable If set to 1, the IPMC provides a trace of SIPL activity on the Payload interface onto the Serial Debug interface Bit [2] Alert Logging Enable If set to 1, the IPMC outputs important alert messages onto the Serial Debug interface Bit [1] Low-level Error Logging Enable If set to 1, the IPMC outputs low-level error/diagnostic messages onto the Serial Debug interface Bit [0] Error Logging Enable If set to 1, the IPMC outputs error/diagnostic messages onto the Serial Debug interface
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Get Hardware Address

The **Get Hardware Address** command reads the hardware address of the IPMC.

Table 7-22: *Get Hardware Address Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	5	Hardware Address

Set Hardware Address

The **Set Hardware Address** command allows overriding of the hardware address read from hardware when the IPMC operates in (Manual) Standalone mode.

Table 7-23: *Set Hardware Address Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Type:	Byte:	Data Field: (continued)
	4	Hardware Address If set to 00, the ability to override the hardware address is disabled NOTE: A hardware address change only takes effect after an IPMC reset. See “Reset IPMC” on page 7-33.
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Get Handle Switch

The **Get Handle Switch** command reads the state of the Hot Swap handle of the IPMC. Overriding of the handle switch state is allowed only if the IPMC operates in (Manual) Standalone mode.

Table 7-24: *Get Handle Switch Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	5	Handle Switch Status 0x00 The handle switch is open 0x01 The handle switch is closed 0x02 The handle switch state is read from hardware

Set Handle Switch

The **Set Handle Switch** command sets the state of the Hot Swap handle switch in (Manual) Standalone mode.

Table 7-25: *Set Handle Switch Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Handle Switch Status 0x00 The handle switch is open 0x01 The handle switch is closed 0x02 The handle switch state is read from hardware
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Get Payload Communication Time-Out

The **Get Payload Communication Time-Out** command reads the payload communication time-out value.

Table 7-26: *Get Payload Communication Time-Out Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	5	Payload Time-out Payload communication time-out measured in hundreds of milliseconds. Thus, the payload communication time-out may vary from 0.1 to 25.5 seconds.

Set Payload Communication Time-Out

The **Set Payload Communication Time-Out** command sets the payload communication time-out value.

Table 7-27: *Set Payload Communication Time-Out Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Payload Time-out Payload communication time-out measured in hundreds of milliseconds. Thus, the payload communication time-out may vary from 0.1 to 25.5 seconds.
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Enable Payload Control

The **Enable Payload Control** command enables payload control from the Serial Debug interface.

Table 7-28: *Enable Payload Control Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Disable Payload Control

The **Disable Payload Control** command disables payload control from the Serial Debug interface.

Table 7-29: *Disable Payload Control Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Reset IPMC

The **Reset IPMC** command allows the payload to reset the IPMC over the SIPL.

Table 7-30: *Reset IPMC Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Reset Type Code 0x00 Cold IPMC reset to the current mode 0x01 Cold IPMC reset to the Normal mode 0x02 Cold IPMC reset to the Standalone mode 0x03 Cold IPMC reset to the Manual Standalone mode 0x04 Reset the IPMC and enter Upgrade mode
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Hang IPMC

The IPMC provides a means to test the watchdog timer support by implementing the **Hang IPMC** command, which simulates firmware hanging by entering an endless loop.

Table 7-31: *Hang IPMC Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Bused Resource

To send a **Bused Resource** command to the carrier controller, the payload uses the **Bused Resource** command of the SIPL.

Table 7-32: *Bused Resource Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Command Types for Carrier Controller to Board 0 Query if board has control of the bus 1 Release requests a board to release control of the bus 2 Force board to release control of bus immediately 3 Bus Free informs board that the bus is available Command Types for Board to Carrier Controller 0 Request to seize control of the bus 1 Relinquish control of the bus, carrier controller can reassign control of bus 2 Notify carrier controller that control of the bused resource has been transferred to this board from another authorized board
	5	Bused Resource ID 0 Metallic Test Bus pair #1 1 Metallic Test Bus pair #2 2 Synch clock group 1 (CLK1A and CLK1B pairs) 3 Synch clock group 2 (CLK2A and CLK2B pairs) 3 Synch clock group 3 (CLK3A and CLK3B pairs)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	5	Status 0 Ack; carrier controller acknowledges that board has control 1 Error; same as Ack, but carrier controller believes board should not have been given control of the resource (optional) 2 Deny; carrier controller denies control of resource by the board

Bused Resource Status

If the IPMC receives a **Bused Resource** command from IPMB-0, it asserts an appropriate event and notifies the payload which uses the **Bused Resource Status** command over the SIPL. When the IPMC receives a Bused Resource Status command, the respective bit in the IPMC status is cleared.

The payload must issue a **Bused Resource Status** command before the payload communication time-out time. If the payload does not issue such a command before the payload communication time-out time, the IPMC sends the 0xC3 completion code (Time-Out) in the appropriate **Bused Resource** command reply.

Table 7-33: *Bused Resource Status Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Command Types for Carrier Controller to Board 0 Query if board has control of the bus (0=In control, 1= No control) 1 Release request a board to release control of the bus (0=Ack, 1=Refused, 2=No control) 2 Force board to release control of bus immediately (0=Ack, 1=No control) 3 Bus Free informs board that the bus is available (0=Accept, 1=Not needed) Command Types for Board to Carrier Controller 0 Request to seize control of the bus (0=Grant, 1=Busy, 2=Defer, 3=Deny) 1 Relinquish control of the bus, carrier controller can reassign control of bus (0=Ack, 1=Error) 2 Notify carrier controller that control of the bused resource has been transferred to this board from another authorized board (0=Ack, 1=Error, 2=Deny)
	5	Bused Resource ID 0 Metallic Test Bus pair #1 1 Metallic Test Bus pair #2 2 Synch clock group 1 (CLK1A and CLK1B pairs) 3 Synch clock group 2 (CLK2A and CLK2B pairs) 4 Synch clock group 3 (CLK3A and CLK3B pairs)
	6	Status 0 Ack; carrier controller acknowledges that board has control 1 Error; same as Ack, but carrier controller believes board should not have been given control of the resource (optional) 2 Deny; carrier controller denies control of resource by the board
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Graceful Reset

The IPMC supports the Graceful Reboot option of the **FRU Control** command. On receiving such a command, the IPMC sets the Graceful Reboot Request bit of the IPMC status, sends a status update notification to the payload, and waits for the **Graceful Reset** command from the payload. If the IPMC receives such a command before the payload communication time-out time, it sends the 0x00 completion code (Success) to the carrier controller. Otherwise the 0xC3 completion code (Time-Out) is sent.

The IPMC does not reset the payload on receiving the **Graceful Reset** command or time-out. If the IPMC participation is necessary, the payload must request the IPMC to perform a payload reset. The **Graceful Reset** command is also used to notify the IPMC about the completion of the payload shutdown sequence.

Table 7-34: *Graceful Reset Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Diagnostic Interrupt Results

The IPMC supports the Issue Diagnostic Interrupt feature of the **FRU Control** command. The payload is notified about a diagnostic interrupt over the SIPL. The payload is expected to return diagnostic interrupt results before the payload communication time-out using the **Diagnostic Interrupt Results** command of the SIPL.

Table 7-35: *Diagnostic Interrupt Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	If the payload responds before the payload communication time-out, the diagnostic interrupt return code is forwarded to the carrier controller as the completion code of the **FRU Control** command response. Otherwise, the 0xC3 completion code (Time-Out) is returned.
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Get Payload Shutdown Time-Out

When the carrier controller commands the IPMC to shut down the payload (i.e. sends the **Set Power Level (0)** command), the IPMC notifies the payload by asserting an appropriate alert and sending an alert notification to the payload. Upon receiving this notification, the payload software is expected to initiate the payload shutdown sequence. After performing this sequence, the payload should send the **Graceful Reset** command to the IPMC over the Payload interface to notify the IPMC that the payload shutdown is complete.

To avoid deadlocks that may occur if the payload software does not respond, the IPMC provides a special time-out for the payload shutdown sequence. If the payload does not send the **Graceful Reset** command within a definite period of time, the IPMC assumes that the payload shutdown sequence is finished, and sends a Module Quiesced Hot Swap event to the ATCA-9305 controller.

Table 7-36: *Get Payload Shutdown Time-Out Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	5:6	Time-Out measured in hundreds of milliseconds, LSB first

Set Payload Shutdown Time-Out

The **Set Payload Shutdown Time-Out** command is defined as follows:

Table 7-37: *Set Payload Shutdown Time-Out Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4:5	Time-Out measured in hundreds of milliseconds, LSB first
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Set Local FRU LED State

The **Set Local FRU LED State** command is used to change the local state of a FRU LED.

Table 7-38: *Set Local FRU LED State Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	FRU Device ID
	5	LED ID 00h Blue LED (Hot Swap) 01h LED 1 (OOS) 02h LED 2 03h LED 3 04h-FEh OEM defined LEDs FFh Lamp Test (all LEDs under management control are addressed)
	6	LED Function 00h LED off override 01h-FAh LED blinking override FBh Lamp Test state Turn on LED specified in byte 3 for the duration specified in byte 5, then return to the highest priority state. FCh LED state restored to Local Control state FDh-FEh reserved FFh LED on override
	7	On Duration LED on-time is measured in tens of milliseconds Lamp Test time in hundreds of milliseconds if byte 4=FBh, time value must be less than 128. Other values when Byte 4=FBh are reserved. Otherwise, this field is ignored and shall be set to 0h.
	8	Color parameter specifies the color of the LED in the local state for multi-color LEDs
	9	If the off-first flag parameter is 0, the on part of the blink cycle of the LED precedes the off part of the cycle. Otherwise, the off part of the blink cycle precedes the on part of the cycle.
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Get Local FRU LED State

The **Get Local FRU LED State** command is used to read the local state of a FRU LED.

Table 7-39: *Get Local FRU LED State Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	FRU Device ID
	5	LED ID 00h Blue LED (Hot Swap) 01h LED 1 (OOS) 02h LED 2 03h LED 3 04h-FEh OEM defined LEDs FFh reserved (all LEDs under management control are addressed)
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	5	Local Control LED Function 00h LED is off (default if Local Control not supported) 01h-FAh LED is blinking Off duration specified by this byte, on duration specified by byte 5 (in tens of milliseconds) FBh-FEh reserved FFh LED is on
	6	Local Control On Duration LED on-time is measured in tens of milliseconds Lamp Test time in hundreds of milliseconds if byte 4=FBh, time value must be less than 128. Other values when Byte 4=FBh are reserved. Otherwise, this field is ignored and shall be set to 0h.
	7	Color parameter specifies the color of the LED in the local state for multi-color LEDs
	8	If the off-first flag parameter is 0, the on part of the blink cycle of the LED precedes the off part of the cycle. Otherwise, the off part of the blink cycle precedes the on part of the cycle.

Update Discrete Sensor

The **Update Discrete Sensor** command is used to change the state of a discrete sensor controlled by the payload.

Table 7-40: *Update Discrete Sensor Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Sensor ID identifies the payload-controlled discrete sensor that has to be updated
	5	Update flags 0 0=sensor initialization is complete 1=sensor is in the initial update state 1:2 reserved, set to 0 3 0=globally disable events from the sensor 1=leave the global event enable bit intact 4 0=globally enable events from the sensor 1=leave the global event enable bit intact 5 0=globally disable sensor scanning 1=leave the global scanning enable bit intact 6 0=globally enable sensor scanning 1=leave the global scanning enable bit intact 7 reserved, set to 0
	6:7	New status LSB and new status MSB are the least and most significant bytes of the new sensor state
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

Update Threshold Sensor

The **Update Threshold Sensor** command is used to change the state of a threshold sensor controlled by the payload.

Table 7-41: *Update Threshold Sensor Command*

Type:	Byte:	Data Field:
Request Data	1:3	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)
	4	Sensor ID parameter identifies the payload-controlled threshold sensor that has to be updated

Type:	Byte:	Data Field: (continued)
Request Data	5	Update flags 0 0=sensor initialization is complete 1=sensor is in the initial update state 1:2 reserved, set to 0 3 0=globally disable events from the sensor 1=leave the global event enable bit intact 4 0=globally enable events from the sensor 1=leave the global event enable bit intact 5 0=globally disable sensor scanning 1=leave the global scanning enable bit intact 6 0=globally enable sensor scanning 1=leave the global scanning enable bit intact 7 reserved, set to 0
	6	New raw reading of the sensor
Response Data	1	Completion Code
	2:4	*PPS IANA Private Enterprise ID*, MS Byte first 0x00400A = 16394 (Pigeon Point Systems)

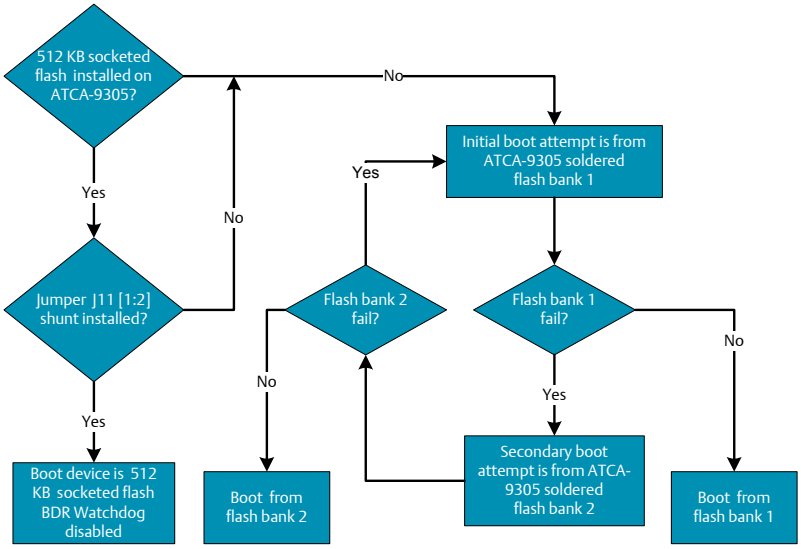
BOOT DEVICE REDIRECTION (BDR)

The IPMC enables the ATCA-9305 to recover from monitor corruption by booting from a redundant copy in another flash device. The mechanism relies on an IPMC software internal watchdog to expire when corrupted code fails to reset the timer. This watchdog begins counting down as soon as the payload is power cycled or reset. If the timer expires (approximately 30 seconds), the boot redirection will activate and the board will reset. Following this automatic reset, IPMC will attempt to boot from the next flash device according to Fig. 7-4. This sequence will continue until a valid boot image clears the watchdog.

The boot redirection order is configurable via the **bootdev** command (see page 9-17). If a shunt is present on J9 [1:2], the ATCA-9305 boots from socket. When forcing boot from the socket, use **bootdev** and reset from the command line to test boot from a flash device. If shunt is not installed on J9 [1:2], the ATCA-9305 follows the default boot redirection shown in Fig. 7-4. Also reference the"Boot Device Redirection" register.

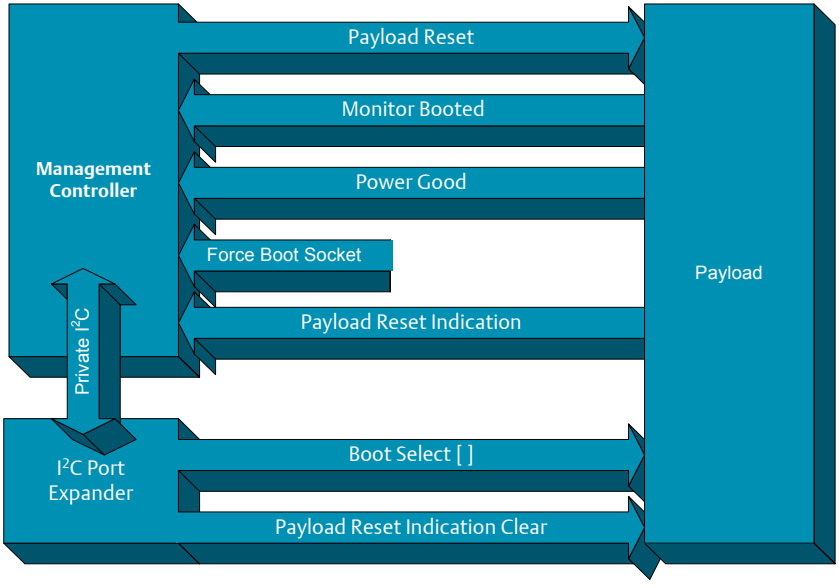
Note: *The System Management IPMC can override the BDFR and swap the flash banks (from 1 to 2, or 2 to 1).*

Figure 7-4: *Boot Device Diagram*



Note: *The Boot Device Redirection mechanism is disabled when booting from the 512 KB socketed flash.*

Figure 7-5: *Boot Redirection Control Diagram*



Management Controller:

The controller provides a signals to reset the payload.

Payload: This provides signals to the controller indicating when the payload has reset for any reason, that the payload is powered, and that the payload has finished its monitor booting sequence. By default, a powered payload enables the watchdog and disables when the payload is not powered.

I^2C Port Expander: The I^2C port expander provides signals to the payload to define the boot device selection (boot select [1 and 2]) and to clear the payload reset indication. The I^2C port expander communicates with the controller via a private I^2C.

Payload Reset: This signal is used by the management controller to reset the payload.

Monitor Booted: This signal indicates to the management controller that a valid monitor image has finished booting and the watchdog can be disabled.

Power Good: This signal indicates to the management controller that the payload is powered. When payload power is applied, the BMC watchdog will start.

Force Boot Socket: If a shunt is present on J9 [1:2], the controller sets the boot location to socket flash with this signal.

Payload Reset Indication:

When reset, this signal is held high by the payload until it is cleared by the IPMC using the payload reset indication clear signal.

Boot Select [ ]: These signals select the boot device.

Payload Reset Indication Clear:

This signal clears the payload reset indication.

MESSAGE LISTENERS

Payload port dynamic control can be implemented via message listeners. The payload can add itself as a message listener to any message destined for the IPMC target either over IPMB-0 or the payload serial interface. When the IPMC receives a subscribed message, the IPMC firmware copies the message into the payload's LUN-10 Receive Message Queue and notifies the payload via an unprintable character (ASCII 07, BELL). The payload receives the message as described in "Message Bridging." The message listener list is only eight entries long. The payload can add/remove/get list at any time.

Note: *The message listener list is not persistent across IPMC reboots.*

Add Message Listener

The **Add Message Listener** command adds a specified Network Function and Command to the Message Listener List. The command returns completion code (0x00) and IANA. If this command does not complete successfully (e.g., due to a full list), it returns 0xCD and IANA.

Table 7-42: *Add Message Listener Command*

Type:	Byte:	Data Field:
Request Data	1:3	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 1 = CD, byte 2 = 65, byte 3 = 00
	4	Network function to add
	5	Command to add
Response Data	1	Completion Code
	2:4	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 2 = CD, byte 3 = 65, byte 4 = 00

Remove Message Listener

The **Remove Message Listener** command removes a specified Network Function and Command from the Message Listener List. The command returns completion code (0x00) and IANA. If this command does not complete successfully (e.g., if the Network Function and Command are not in the list), it returns 0xCD and IANA.

Table 7-43: *Remove Message Listener Command*

Type:	Byte:	Data Field:
Request Data	1:3	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 1 = CD, byte 2 = 65, byte 3 = 00
	4	Network function to remove
	5	Command to remove
Response Data	1	Completion Code
	2:4	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 2 = CD, byte 3 = 65, byte 4 = 00

Get Message Listener List

The **Get Message Listener List** command returns the entire list of subscribed Message Listeners. The command returns completion code (0x00) and IANA.

Table 7-44: *Get Message Listener List Command*

Type:	Byte:	Data Field:
Request Data	1:3	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 1 = CD, byte 2 = 65, byte 3 = 00
Response Data	1	Completion Code
	2:4	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 2 = CD, byte 3 = 65, byte 4 = 00
	5	Network function for listener 0
	6	Command for listener 0
	7	Network function for listener 1
	8	Command for listener 1
	9	Network function for listener 2
	10	Command for listener 2
	11	Network function for listener 3
	12	Command for listener 3
	13	Network function for listener 4
	14	Command for listener 4
	15	Network function for listener 5
	16	Command for listener 5
	17	Network function for listener 6
	18	Command for listener 6
	19	Network function for listener 7
	20	Command for listener 7

SYSTEM FIRMWARE PROGRESS SENSOR

The **Update System Firmware Progress Sensor** command sets the values for the Firmware Progress Sensor using sensor codes from the IPMI Intelligent Platform Management Interface Specification, specifically (System Firmware Progress" within Table 42-3 in Section 42.2 "Sensor Type Codes and Data."

The command returns 0xC0 when the IPMC is busy and will retry until the command is successful. If this command returns 0xCC, the sensor ID is invalid. There is only one sensor on the board, so the sensor ID should always be "0". When updated, the shelf manager is notified.

Table 7-45: *Update System Firmware Progress Sensor Command*

Type:	Byte:	Data Field:
Request Data	1:3	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 1 = CD, byte 2 = 65, byte 3 = 00
	4	0 (The sensor ID)
	5	Flags: reserved to 0
	6	Offset in specification Valid offsets: 0, 1, 2
	7	Event Data 2; content to be added into the second byte of event data per the IPMI specification
Response Data	1	Completion Code
	2:4	*Emerson Network Power, Embedded Computing Inc. IANA Private Enterprise ID* 0x0065CD = 26061 (Emerson Network Power, Embedded Computing Inc.) LSB Byte first:byte 2 = CD, byte 3 = 65, byte 4 = 00

ENTITIES AND ENTITY ASSOCIATIONS

The AdvancedTCA specification (see *PICMG Engineering Change Notice 3.0* listed in Table 1-2) uses Entity IDs and Instances to describe physical components associated with FRUs. Device-relative Entities are unique to a specific IPMC and are referenced as follows in the specification:

r(<ipmb>,<lun>,<Entity ID>,<Entity Instance - 60>)

Using this terminology, a ATCA-9305 installed in Logical Slot 1 has the following description in Fig. 7-6.

Figure 7-6: *IPMB Entity Structure*

FRU 0 r(82, 0, A0, 0)
- Inflow Temp
- Outflow Temp
- Hot Swap
- IPMB Physical
- BMC Watchdog
- F/W Progress
- SDRAM POST
- IIC Bus POST
- Flash POST
- EthSwitch POST
- Version change
- Async Pld Rst
- Payload Power

r(82, 0, 03, 0) - Cavium 1
- Cavium 1 Temp
- Cav1 SDRAM POST
- Cav1 IIC POST
- Cav1 Boot

r(82, 0, 03, 1) - Cavium 2
- Cavium 2 Temp
- Cav2 SDRAM POST
- Cav2 IIC POST
- Cav2 Boot

r(82, 0, 14, 0) - Power Module
- -48V
- -48V Curr
- -48V Src A
- -48V Src B
- +3.3V Mgmt
- +12V Payload
- +12V Curr

FRU 1 r(82, 0, C0, 1) RTM
- RTM Hot Swap

SENSORS AND SENSOR DATA RECORDS

The ATCA-9305 implements a number of sensors as described in the following tables. All values are hexadecimal.

Table 7-46: *IPMI Threshold Sensors*

Name:	Sensor Type:	Event Reading Type:	Enity ID:	Entity Instance:	Event Gen:
Inflow Temp	Temperature = 01	Threshold = 01	0xA0	0x60	Yes
Outflow Temp	Temperature = 01	Threshold = 01	0xA0	0x60	Yes
Cavium 1 Temp	Temperature = 01	Threshold = 01	0x03	0x60	Yes
Cavium 2 Temp	Temperature = 01	Threshold = 01	0x03	0x61	Yes
-48V	Voltage = 02	Threshold = 01	0x14	0x60	Yes
-48V Curr	Current = 03	Threshold = 01	0x14	0x60	Yes
-48V Src A	Voltage = 02	Threshold = 01	0x14	0x60	Yes
-48V Src B	Voltage = 02	Threshold = 01	0x14	0x60	Yes
+3.3V Mgmt	Voltage = 02	Threshold = 01	0x14	0x60	Yes
+12V Payload	Voltage = 02	Threshold = 01	0x14	0x60	Yes
+12V Curr	Current = 03	Threshold = 01	0x14	0x60	Yes

Table 7-47: *IPMI Discrete Sensors*

Name:	Sensor Type:	Event Reading Type:	Enity ID:	Entity Instance:	Event Gen:
Hot Swap	Hot Swap = F0	Sensor specific discrete = 6F	0xA0	0x60	Yes
RTM Hot Swap	Hot Swap = F0	Sensor specific discrete = 6F	0xC0	0x61	Yes
IPMB Physical	IPMB Link = F1	Sensor specific discrete = 6F	0xA0	0x60	Yes
BMC Watchdog	Watchdog2 = 23	Sensor specific discrete = 6F	0xA0	0x60	Yes
F/W Progress	System Firmware Progress = 0F	Sensor specific discrete = 6F	0xA0	0x60	Yes
SDRAM POST	Memory = 0C	Sensor specific discrete = 6F	0xA0	0x60	Yes
IIC Bus POST	Processor = 07	Predictive-failure Discrete = 04	0xA0	0x60	Yes
Flash POST	Memory = 0C	Sensor specific discrete = 6F	0xA0	0x60	Yes
EthSwitch POST	Chip Set	Predictive-failure Discrete = 04	0xA0	0x60	Yes
Cav1 SDRAM POST	Memory = 0C	Sensor specific discrete = 6F	0x03	0x60	Yes

Name:	Sensor Type:	Event Reading Type:	Enity ID:	Entity Instance:	Event Gen:
Cav1 IIC POST	Processor = 07	Predictive-failure Discrete = 04	0x03	0x60	Yes
Cav1 Boot	Processor = 07	Predictive-failure Discrete = 04	0x03	0x60	Yes
Cav2 SDRAM POST	Memory = 0C	Sensor specific discrete = 6F	0x03	0x61	Yes
Cav2 IIC POST	Processor = 07	Predictive-failure Discrete = 04	0x03	0x61	Yes
Cav2 Boot	Processor = 07	Predictive-failure Discrete = 04	0x03	0x61	Yes
Version change	Version Change	Sensor specific discrete = 6F	0xA0	0x60	Yes
Async Pld Rst	Power Supply = 08	Digital Discrete = 03	0xA0	0x60	Yes
Payload Power	Power Supply = 08	Digital Discrete = 03	0xA0	0x60	Yes

The IPMC implements a Device Sensor Data Record (SDR) Repository that contains SDRs for the IPMC, the FRU device, and each sensor. A system management controller may use the Get Device SDR command to read the repository and dynamically discover the capabilities of the board. Refer to the IPMI specification (listed in Table 1-2) for more information on using Sensor Data Records and the Device SDR Repository.

Under certain circumstances, some sensors connected to the IPMC can generate Event Messages for the system management controller. To enable these messages, the system management controller must send a **Set Event Receiver** command to the IPMC, along with the address of the Event Receiver. Table 7-48 shows the format of an Event Message:

Table 7-48: *Event Message Format*

Byte:[1]	Field:	Description:
0	RsSA	Responder's Slave Address (Address of Event Receiver)
1	NetFn/RsLUN	Net Function Code (0x04) in upper 6 bits; Responder's LUN in lower 2 bits
2	Chk 1	Checksum #1
3	RqSA	Requester's Slave Address (Address of our board on IPMB)
4	RqSeq/RqLUN	Request Sequence number in upper 6 bits; Requester's LUN in lower 2 bits
5	Cmd	Command (Always 0x02 for event message)
6	EvMRev	Event Message Revision (0x04 for IPMI 1.5)
7	Sensor Type	Indicates event class or type of sensor that generated the message
8	Sensor Number	A unique number indicating the sensor that generated the message

Byte:[1]	Field:	Description: (continued)
9	Event Dir/Event Type	Upper bit indicates direction (0 = Assert, 1= Deassert); Lower 7 bits indicate type of threshold crossing or state transition
10	Event Data 0	Data for sensor and event type
11	Event Data 1	(Optional) Data for sensor and event type
12	Event Data 2	(Optional) Data for sensor and event type
13	Chk2	Checksum #2

1. Each byte has eight bits.

Event-generating sensors with a Threshold Event/Reading Type (0x01) initiate an event message when a sensor reading crosses the defined threshold. The default thresholds for a particular sensor are retrieved by sending the IPMC a **Get Sensor Thresholds** command. The system management controller must send the IPMC a **Get Sensor Reading** command to retrieve the current sensor reading. Refer to the IPMI specification listed in Table 1-2 for complete details on using these commands.

FRU INVENTORY

The IPMC stores Field Replaceable Unit (FRU) information in its boot memory (SROM). The data structure contains information such as the product name, part number, serial number, manufacturing date, and E-keying information. Refer to the IPMI specification for complete details on the FRU data structure. Table 7-49 lists the general contents of the ATCA-9305's FRU information:

Table 7-49: *FRU Definition*

Item:	Description:
Common Header	
Version	Version number of the overall FRU data structure defined by the IPMI FRU specification
Internal Use Area	
Version	Version number of the Internal Use Area data structure defined by the IPMI FRU specification
Internal Use Size	0x100 bytes are allocated for customer use in this area
Board Information Area	
Version	Version number of the Board Information Area data structure defined by the IPMI FRU specification
Language Code	0x01 = English
Manufacturing Date/Time	Variable, expressed as the number of minutes since 12:00 AM on January 1, 1996
Board Manufacturer	"Emerson"
Board Product Name	"ATCA-9305"

Item:	Description: (continued)
Board Serial Number	Variable, formatted as “730-XXXX”
Board Part Number	Variable, formatted as “10XXXXXX-YY-Z”
FRU File ID	Variable, for example: “fru-info.inf”
Product Information Area	
Version	Version number of the Product Information Area data structure defined by the IPMI FRU specification
Language Code	0x01 = English
Manufacturer Name	“Emerson”
Product Name	“ATCA-9305”
Product Part/Model Number	Variable, formatted as “10XXXXXX-YY-Z”
Product Version	Not used, same information is provided by the part number
Product Serial Number	Variable, formatted as “730-XXXX”
Asset Tag	Not Used
FRU File ID	Variable, for example: “fru-info.inf”
MultiRecord Area	
E-Keying records	See “E-Keying”
Maximum Internal Current	“12.5 Amps”

E-KEYING

This section details the interfaces governed by E-keying and the protocols they support. Specifically, this includes the interfaces implemented by this product and the E-keying definition that corresponds to each interface.

The IPMC supports E-keying for the ATCA-9305 per PICMG® ATCA 3.0, Revision 2.0 and PICMG 3.1, Revision 1.0 specifications The E-keying information is stored in the ATCA Point-to-Point Connectivity Record located in the Multi-Record area of the FRU Inventory Information (see page 7-50). The ATCA Point-to-Point Connectivity Record contains a Channel Descriptor list, where each Link Descriptor details one type of point-to-point protocol supported by the referenced channels.

The ATCA channel descriptors define the ATCA channels implemented on a module. Each channel has an arbitrary set of up to four ports. Channel descriptors map physical ports to logical entities known as lanes, see Table 7-50.

Note: *Certain Ethernet core switch and fat pipe switch module GbE switch ports are disabled due to lack of e-keying support in the monitor.*

Base Point-to-Point Connectivity

The ATCA-9305 supports two 10/100/1000BASE-T ports on Base Interface Channels 0 and 1, and two 10 GbE XAUI ports to the Fabric channels. Depending on the board configuration, either two or six 10 GbE XAUI ports route to the optional rear transition module (RTM). Table 7-50 shows the Point-to-point Connectivity Record Link Descriptors for the ATCA-9305.

Note: *For actual Point-to-Point connectivity Records for your configuration, query the IPMI controller.*

Table 7-50: *Link Description*

Field:	Value:[1]	Description:
Link Descriptor	000100000000b	Port 0 Enabled; Base Interface; Channel 1
Link Type	01h	PICMG 3.0 Base Interface 10/100/1000BASE-T
Link Type Extension	000b	
Link Grouping ID	00h	Independent Channel
Link Designator	000100000001b	Port 0 Enabled; Base Interface; Channel 2
Link Type	01h	PCIMG 3.0 Base Interface 10/100/1000BASE-T
Link Type Extension	0000b	
Link Grouping ID	00h	Independent Channel
Link Designator	000110000001b	Port 0 Enabled; Update Channel Interface; Channel 1
Link Type	01h	PICMG 3.1 Ethernet Fabric Interface
Link Type Extension	0000b	Fixed 1000BASE-BX
Link GroupingID	00h	Independent Channel

1. h = hexadecimal, b = binary

HPM.1 FIRMWARE UPGRADE

The ATCA-9305 IPMC firmware supports a reliable field upgrade procedure compliant with the HPM.1 specification. The prominent features of the firmware upgrade procedure are:

- The upgrade can be performed either over the payload serial interface or IPMB-0.
- The upgrade procedure is performed while the ATCA-9305 is online and operating normally.
- The upgrades are reliable. A failure in the download (error or interruption) does not disturb the ATCA-9305's ability to continue using the "old" firmware or its ability to restart the download process.
- The upgrades are reversible. The ATCA-9305 IPMC automatically reverts back to the previous firmware if there is a problem when first running the new code, and can be reverted manually using the HPM.1-defined Manual Rollback command.

HPM.1 Reliable Field Upgrade Procedure

The HPM.1 upgrade procedure is managed by a utility called the Upgrade Agent. The *Impitool* utility is used as an Upgrade Agent for upgrading the ATCA-9305 IPMC firmware.

The Upgrade Agent communicates with the IPMC firmware via the payload serial interface or IPMC-0, and uses the AdvancedTCA commands that are described in the HPM.1 specification for upgrading the firmware. Updated firmware is packed into an image formatted in compliance with the HPM.1 specification. That image is used by Upgrade Agent to prepare and upgrade the IPMC firmware. The HPM.1 upgrade procedure includes the following steps:

Preparation: This step erases the region in the flash memory where the component image will be written.

Component Upload: This step is designed to upload the component image via IPMB or payload interface and write it into the flash memory.

Component Activation: This step activates the previously upgraded component. This step can be deferred and performed later.

For more details, refer to the HPM.1 specification listed in Table 1-2.

IPMC HEADERS

This JTAG header (JP1) is available for in-system programming of the CPLD.

Table 7-51: *IPMP CPLD JP1 Pin Assignments*

Pin:	Signal:	Direction:	Pin:	Signal:
1	CPLD_TCK	out	2	ground
3	CPLD_TDI	in	4	3_3V (fused)
5	CPLD_TMS	out	6	no connect
7	no connect	–	8	no connect
9	CPLD_TDO	out	10	ground

The EIA-232 debug serial port is accessible via the mini-B USB connector P4. Default port settings are: 115200 baud (optional 9600), 8 data bits, 1 stop bit, no parity, no flow control.

Table 7-52: *IPMP EIA-232 P4 Pin Assignments*

Pin:	Signal:	Pin:	Signal:
1	no connect	2	IPMP_RS_232_Rx
3	IPMP_RS_232_Tx	4	no connect
5	ground	6	ground
7	ground		

(blank page)

Back Panel Connectors

There are multiple connectors on the ATCA-9305, reference Fig. 2-2 for their location. The back panel connectors, Zones 1 through 3, are described in this chapter. Whether individual back panel connectors are populated on the ATCA-9305 depends on the specific product configuration.

ZONE 1

Connector P10 provides the AdvancedTCA Zone 1 power (dual redundant -48 VDC) and system management connections. Four levels of sequential mating provide proper functionality during live insertion or extraction, see Table 8-1.

Figure 8-1: *Zone 1 Connector, P10*

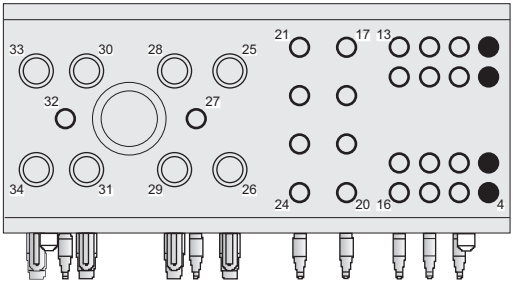


Table 8-1: *Zone 1 Connector, P10 Pin Assignments*

Pin:	Signal:	Insertion Sequence:
1	reserved	NA
2	reserved	NA
3	reserved	NA
4	reserved	NA
5	HA0	third
6	HA1	third
7	HA2	third
8	HA3	third
9	HA4	third
10	HA5	third
11	HA6	third
12	HA7 (odd parity bit)	third
13	IPMBA_SCL	third
14	IPMBA_SDA	third
15	IPMBB_SCL	third
16	IPMBB_SDA	third
17	no connect	third

Pin:	Signal:	Insertion Sequence:
18	no connect	third
19	no connect	third
20	no connect	third
21	no connect	third
22	no connect	third
23	no connect	third
24	no connect	third
25	P10_CHS_GND	first
26	Logic ground	first
27	ENABLE_B	fourth
28	-48RTNA	first
29	-48RTNB	first
30	no connect	first
31	no connect	first
32	ENABLE_ A	fourth
33	-48A	second
34	-48B	third

ZONE 2

Zone 2 (ZD) defines backplane connector J23, which supports the data transport interface. The Zone 2 connector array supports four interfaces to the AdvancedTCA backplane:

- Base Node Interface (J23) supports two Base channels (10/100/1000 BASE-T)
- Fabric Interface (J23) supports two Fabric channels (10GbE)

Figure 8-2: *Zone 2 and 3 Connectors; J23, J30-J31*

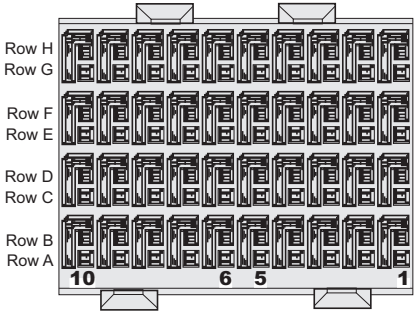


Table 8-2: *Zone 2 Connector, J23 Pin Assignments*

Row:	Interface:	AB:		CD:		EF:		GH:	
1	Fabric Channel 2	TX2+	TX2-	RX2+	RX2-	TX3+	TX3-	RX3+	RX3-
2		TX0+	TX0-	RX0+	RX0-	TX1+	TX1-	RX1+	RX1-
3	Fabric Channel 1	TX2+	TX2-	RX2+	RX2-	TX3+	TX3-	RX3+	RX3-
4		TX0+	TX0-	RX0+	RX0-	TX1+	TX1-	RX1+	RX1-
5	Base Channel 1	TRD0+	TRD0-	TRD1+	TRD1-	TRD2+	TRD2-	TRD3+	TRD3-
6	Base Channel 2	TRD0+	TRD0-	TRD1+	TRD1-	TRD2+	TRD2-	TRD3+	TRD3-
7-10	na		no connect						

ZONE 3

These optional Zone 3 type A connectors, J30, J31, and J33, support a Rear Transition Module (RTM). I/O signals are routed through Zone 3 connectors to the RTM to allow servicing the ATCA-9305 without using cable assemblies. Connectors J30 and J31 use the same ZD connector as Zone 2. See Fig. 8-3 for the J33 connector.

Table 8-3: *Zone 3 Connector, J30 Pin Assignments*

	A:	B:	C:	D:	E:	F:	G:	H:
1	RTM_10G1_RX0_P	RTM_10G1_RX0_N	PQ_PCIE_RXD3_P	PQ_PCIE_RXD3_N	RTM_10G2_RX0_P	RTM_10G2_RX0_N	PQ_PCIE_TXD3_P	PQ_PCIE_TXD3_N
2	RTM_10G1_RX1_P	RTM_10G1_RX1_N	PQ_PCIE_RXD2_P	PQ_PCIE_RXD2_N	RTM_10G2_RX1_P	RTM_10G2_RX1_N	PQ_PCIE_TXD2_P	PQ_PCIE_TXD2_N
3	RTM_10G1_RX2_P	RTM_10G1_RX2_N	PQ_PCIE_RXD1_P	PQ_PCIE_RXD1_N	RTM_10G2_RX2_P	RTM_10G2_RX2_N	PQ_PCIE_TXD1_P	PQ_PCIE_TXD1_N
4	RTM_10G1_RX3_P	RTM_10G1_RX3_N	PQ_PCIE_RXD0_P	PQ_PCIE_RXD0_N	RTM_10G2_RX3_P	RTM_10G2_RX3_N	PQ_PCIE_TXD0_P	PQ_PCIE_TXD0_N
5	RTM_10G1_TX0_P	RTM_10G1_TX0_P	PCIE_REFCLKF_P	PCIE_REFCLKF_N	RTM_10G2_TX0_P	RTM_10G2_TX0_N	no connect	no connect
6	RTM_10G1_TX1_P	RTM_10G1_TX1_N	no connect	no connect	RTM_10G2_TX1_P	RTM_10G2_TX1_N	no connect	no connect
7	RTM_10G1_TX2_P	RTM_10G1_TX2_N	no connect	no connect	RTM_10G2_TX2_P	RTM_10G2_TX2_N	no connect	no connect
8	RTM_10G1_TX3_P	RTM_10G1_TX3_N	no connect	no connect	RTM_10G2_TX3_P	RTM_10G2_TX3_N	no connect	no connect
9	RTM_ID3	RTM_ID2	no connect	no connect	RTM_GPIO3	RTM_GPIO2	RTM_GPIO7	RTM_GPIO6
10	RTM_ID1	RTM_ID0	SW_MDC	SW_MDIO	RTM_GPIO1	RTM_GPIO0	RTM_GPIO5	RTM_GPIO4

Table 8-4: *Zone 3 Connector, J31 Pin Assignments*

	A:	B:	C:	D:	E:	F:	G:	H:
1	RTM_10G3 _RX0_P	RTM_10G3 _RX0_N	RTM_10G5 _RX0_P	RTM_10G5 _RX0_N	RTM_10G4 _RX0_P	RTM_10G4 _RX0_N	RTM_10G6 _RX0_P	RTM_10G6 _RX0_N
2	RTM_10G3 _RX1_P	RTM_10G3 _RX1_N	RTM_10G5 _RX1_P	RTM_10G5 _RX1_N	RTM_10G4 _RX1_P	RTM_10G4 _RX1_N	RTM_10G6 _RX1_P	RTM_10G6 _RX1_N
3	RTM_10G3 _RX2_P	RTM_10G3 _RX2_N	RTM_10G5 _RX2_P	RTM_10G5 _RX2_N	RTM_10G4 _RX2_P	RTM_10G4 _RX2_N	RTM_10G6 _RX2_P	RTM_10G6 _RX2_N
4	RTM_10G3 _RX3_P	RTM_10G3 _RX3_N	RTM_10G5 _RX3_P	RTM_10G5 _RX3_N	RTM_10G4 _RX3_P	RTM_10G4 _RX3_N	RTM_10G6 _RX3_P	RTM_10G6 _RX3_N
5	RTM_10G3 _TX0_P	RTM_10G3 _TX0_N	RTM_10G5 _TX0_P	RTM_10G5 _TX0_N	RTM_10G4 _TX0_P	RTM_10G4 _TX0_N	RTM_10G6 _TX0_P	RTM_10G6 _TX0_N
6	RTM_10G3 _TX1_P	RTM_10G3 _TX1_N	RTM_10G5 _TX1_P	RTM_10G5 _TX1_N	RTM_10G4 _TX1_P	RTM_10G4 _TX1_N	RTM_10G6 _TX1_P	RTM_10G6 _TX1_N
7	RTM_10G3 _TX2_P	RTM_10G3 _TX2_N	RTM_10G5 _TX2_P	RTM_10G5 _TX2_N	RTM_10G4 _TX2_P	RTM_10G4 _TX2_N	RTM_10G6 _TX2_P	RTM_10G6 _TX2_N
8	RTM_10G3 _TX3_P	RTM_10G3 _TX3_N	RTM_10G5 _TX3_P	RTM_10G5 _TX3_N	RTM_10G4 _TX3_P	RTM_10G4 _TX3_N	RTM_10G6 _TX3_P	RTM_10G6 _TX3_N
9	no connect	no connect	no connect	no connect	no connect	no connect	no connect	no connect
10	no connect	no connect	no connect	no connect	no connect	no connect	no connect	no connect

Figure 8-3: *Zone 3 Connector, J33*

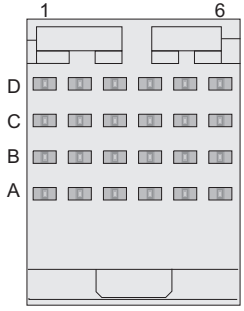


Table 8-5: *Zone 3 Connector, J33 Pin Assignments*

Pin:	A:	B:	C:	D:
1	RTM_ENABLE	RTM_PS1_CONN*	PQ_CONSOLE_RX_M	RTM_PB_RST*
2	RTM_PP_PWRGD	RTM_HS_LED	PQ_CONSOLE_TX_M	RTM_E_HANDLE
3	RTM_MP_PWRGD	IPMB_RTM_SCL_BUFF	no connect	RTM_RST*
4	no connect	IPMB_RTM_SDA_BUFF	3_3V_MP_RTM	3_3V_MP_RTM
5	ground	ground	ground	ground
6	12V_RTM	12V_RTM	12V_RTM	12V_RTM

Management Processor Monitor

The ATCA-9305 monitor is based on the Embedded PowerPC Linux Universal Boot (U-Boot) Project program, available under the GNU General Public License (GPL). For instructions on how to obtain the source code for this GPL program, please visit http://www.emersonembeddedcomputing.com/post-sales_support/218, send an e-mail to support@artesyncp.com, or call Emerson at (800) 327-1251.

This chapter describes the monitor's basic features, operation, and configuration sequences. This chapter also serves as a reference for the monitor commands and functions.

COMMAND-LINE FEATURES

The ATCA-9305 monitor uses a command-line interface with the following features:

Auto-Repeat: After entering a command, you can re-execute it simply by pressing the ENTER or RETURN key.

Command History: Recall previously entered commands using the up and down arrow keys.

TFTP Boot: You can use the TFTP protocol to load application images via Ethernet into the ATCA-9305's memory.

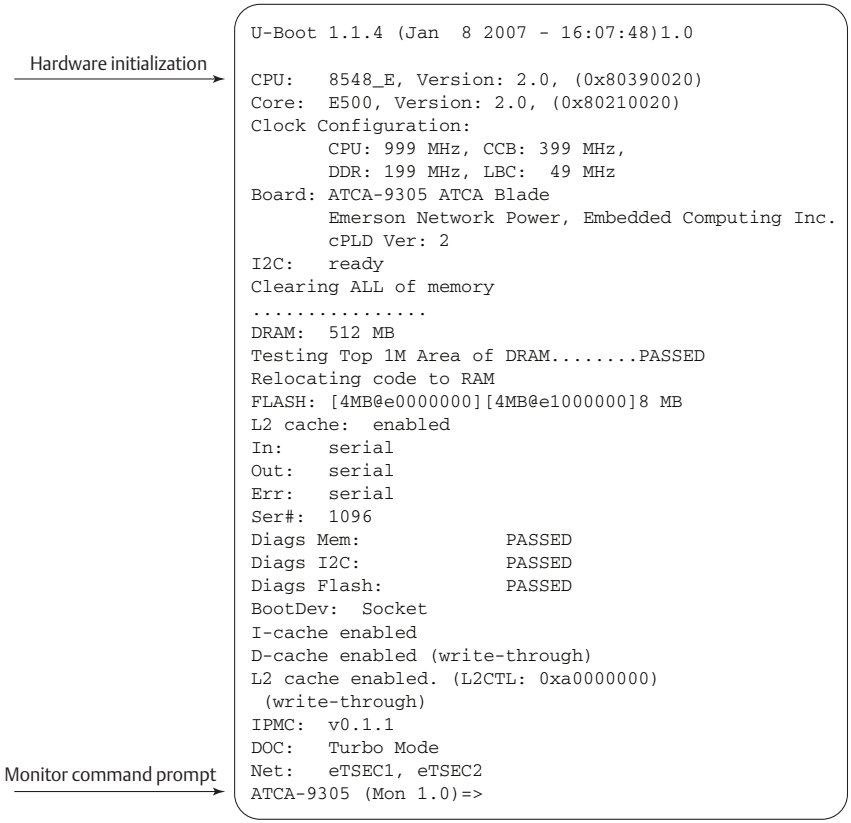
Auto-Boot: You can store specific boot commands in the environment to be executed automatically after reset.

Flash Programming: You can write application images into flash via the U-Boot command line. The upper 1 MB at the base of flash and 128 KB of each flash bank is reserved for the monitor and environment variables (see "MPC8548 Memory Map"). One megabyte is reserved at the second bank of flash. The moninit command will load both banks of flash (see "moninit" on page 9-22) with the monitor and default environment variables.

At power-up or after a reset, the monitor runs diagnostics and reports the results in the start-up display, see Fig. 9-1. During the power-up sequence, the monitor configures the board according to the environment variables (see "MPC8548 Environment Variables" on page 9-26). If the configuration indicates that autoboot is enabled, the monitor attempts to load the application from the specified device. If the monitor is not configured for autoboot or a failure occurs during power-up, the monitor enters normal command-line mode. Also, the optional "e-keying" environment variable enables connections at power-up, for debug purposes only, to the Update Channel and payload ports that go off the ATCA-9305. See Table 9-7 for more information.

The monitor command prompt in Fig. 9-1 is the result of a successful hardware boot of the ATCA-9305.

Figure 9-1: *Example MPC8548 Monitor Start-up Display*



This prompt is also displayed as an indication that the monitor has finished executing a command or function invoked at the command prompt (except when the command loads and jumps to a user application). The hardware product name (ATCA-9305), and current software version number are displayed in the prompt.

Prior to the console port being available, the monitor will display a four-bit hexadecimal value on LED1 through LED4 to indicate the power-up status (see Table 9-1). See Fig. 2-4 for the debug LED locations. In the event of a specific initialization error, the LED pattern will be displayed and the board initialization will halt.

Table 9-1: *Debug LED Codes*

LED Code:	Power-up Status:	LED Value:
BOARD_PRE_INIT	start booting, setup BATs done	0x01
SERIAL_INIT	console init done	0x02

LED Code:	Power-up Status:	LED Value:
CHECKBOARD	get processor and bus speeds done	0x03
SDRAM_INIT	RAM / ECC init done	0x04
AFTER_RELOC	U-Boot relocated to RAM done	0x05
MISC_R	final init including Ethernet done	0x06
GONE_TO_PROMPT	–	0x00

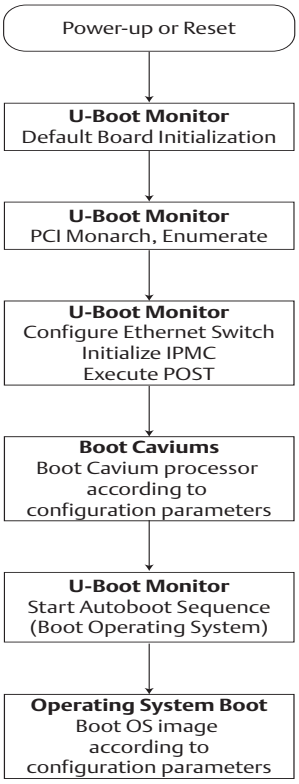
BASIC OPERATION

The monitor performs various configuration tasks upon power-up or reset. This section describes the monitor operation during initialization of the ATCA-9305 board. The flow-chart (see Fig. 9-2) illustrates the power-up and global reset sequence (bold text indicates environment variables).

Power-up/Reset Sequence

The ATCA-9305 monitor follows the boot sequence in Fig. 9-2 before auto-booting the operating system or application software. At power-up or board reset, the monitor performs hardware initialization, diagnostic routines, autoboot procedures, free memory initialization, and if necessary, invokes the command line. See Fig. 3-4 for the Cavium CN5860 processor boot sequence. See Table 9-6 for default environment variables settings.

Figure 9-2: *Power-up/Reset Sequence Flowchart*



POST Diagnostic Results

The ATCA-9305 Power-On Self-Test (POST) diagnostic results are stored as a 32-bit value in I^2C NVRAM at the offset 0x07F0-0x07FF. Each bit indicates the result of a specific test, therefore this field can store the results of up to 32 diagnostic tests. Table 9-2 assigns the bits to specific tests.

Table 9-2: *POST Diagnostic Results—Bit Assignments*

Bit:	Diagnostic Test:	Description:	Value:
0	SDRAM	Verify address and data lines are intact	0 Passed the test 1 Failure detected
1	Flash	Verify size and initialization of soldered flash	
2	I^2C	Verify all local I^2C devices are connected to the I^2C bus	
3	Ethernet Switch	Verify PCI communication with switch	
4	Reserved		
5	PCIe Time-out	PCIe enumeration skipped by user	
6	DOC Embedded Flash Drive (EFD)	Verify presence and ability to access configuration space of DOC	
7	Cavium 1 Presence	Verify presence and ability to communicate via PCI bus with Cavium 1	
8	Cavium 2 Presence	Verify presence and ability to communicate via PCI bus with Cavium 2	
9-31	Reserved		

Monitor SDRAM Usage

Monitor SDRAM usage is typically around 1 MB for monitor code and stack support. Please note that the monitor stack grows downward from below where the monitor code resides (in the upper 512 KB). The monitor C stack will typically not grow beyond 512 KB, therefore the upper 1 MB of SDRAM is reserved for monitor use.

Note: *The monitor has the ability to preserve (not overwrite) areas of memory defined by the pram environment variable.*

Caution: Any writes to these areas can cause unpredictable operation of the monitor.

MONITOR RECOVERY AND UPDATES

This section describes how to recover and/or update the monitor, given one or more of the following conditions:

- If there is no console output, the monitor may be corrupted and need recovering (see the “Recovering the Monitor” section).
- If the monitor still functions, but is not operating properly, then you may need to reset the environment variables (see the “Resetting Environment Variables” section).
- If you are having Ethernet problems in the monitor, you may need to set the serial number, since the MAC address is calculated from the serial number variable.

Recovering the Monitor

1. Make sure that a monitor ROM device is installed in the PLCC socket on the ATCA-9305.
2. Verify there is a shunt on J9, across pins 1 and 2.
3. Issue the following command, where serial_number is the board's serial number, at the monitor prompt:

```
ATCA-9305 (1.0) => moninit serial_number
```

moninit will also reset environment variables to the default state.

4. To boot from soldered flash, power down the board and remove the shunt from J9, pins 1 and 2.

The monitor always resides in the top 512 KB block of NOR flash (banks 1 and 2) as shown in Table 9-3.

Table 9-3: *Monitor Address per Flash Device*

Address Range (hex):	Device:
F3F8,0000-F400,0000	Monitor Location in Flash Bank2 (4 MB)
F3B8,0000-F3C0,0000	Monitor Location in Flash Bank1 (4 MB)
F3B7,0000-F3B7,1000	Environment Variables
F3F7,0000-F3F7,1000	Redundant Environment Variables

Resetting Environment Variables

To restore the monitor's standard environment variables, execute the following commands and insert the appropriate *data* in the italicized fields:

```
ATCA-9305 (1.0) => moninit serial_number noburn
```

Note: *Press the 's' key on the keyboard during reset to force the default environment variables to be loaded. See "MPC8548 Environment Variables" for more information.*

Optionally, save your settings:

```
ATCA-9305 (1.0) => saveenv
```

Updating the Monitor via TFTP

To update the monitor via TFTP, ensure that an appropriate VLAN is set up in the Ethernet switch (see the *ATCA-9305 Quick Start Guide*, #10009110-xx) and execute the following commands, inserting the appropriate *data* in the italicized fields:

If necessary, edit your network settings:

```
ATCA-9305 (1.0) => setenv ipaddr 192.168.1.100
ATCA-9305 (1.0) => setenv gatewayip 192.168.1.1
ATCA-9305 (1.0) => setenv netmask 255.255.255.0
ATCA-9305 (1.0) => setenv serverip 10.64.16.168
```

```
ATCA-9305 (1.0) => setenv ethport eTSEC1
```

Optionally, save your settings:

```
ATCA-9305 (1.0) => saveenv
```

TFTP the new monitor (binary) image to memory location 0x100000:

```
ATCA-9305 (1.0) => tftpboot 100000 path_to_file_on_tftp_server
```

Update the monitor:

```
ATCA-9305 (1.0) => moninit serial_number 100000
```

If **moninit( )** fails, burn the new monitor to a ROM and follow the recovery steps in the "Recovering the Monitor" section.

MONITOR COMMAND REFERENCE

This section describes the syntax and typographic conventions for the ATCA-9305 monitor commands. Subsequent sections in this chapter describe individual commands, which fall into the following categories: boot, memory, flash, environment variables, test, and other commands.

Command Syntax

The monitor uses the following basic command syntax:

```
<Command> <argument 1> <argument 2> <argument 3>
```

- The command line accepts three different argument formats: string, numeric, and symbolic. All command arguments must be separated by spaces with the exception of argument flags, which are described below.
- Monitor commands that expect numeric arguments assume a hexadecimal base.
- All monitor commands are case sensitive.
- Some commands accept flag arguments. A flag argument is a single character that begins with a period (.). There is no white space between an argument flag and a command. For example, **md.b 80000** is a valid monitor command, while **md .b 80000** is not.
- Some commands may be abbreviated by typing only the first few characters that uniquely identify the command. For example, you can type **h** instead of **help**. However, commands cannot be abbreviated when accessing online help. You must type **help** and the full command name.

Command Help

Access all available monitor commands by pressing the **?** key or entering **help**. Access the monitor online help for individual commands by typing **help <***command***>**. The full command name must be entered to access the online help.

Typographic Conventions

In the following command descriptions, text in `Courier` shows the command format. Square brackets [ ] enclose optional arguments, and angled brackets < > enclose required arguments. *Italic* type indicates a variable or field that requires input.

BOOT COMMANDS

The boot commands provide facilities for booting application programs and operating systems from various devices.

bootd

Execute the command stored in the *bootcmd* environment variable.

Definition: `bootd`

bootelf

The **bootelf** command boots from an ELF image in memory, where *address* is the load address of the ELF image.

Definition: `bootelf [ address ]`

bootm

The **bootm** command boots an application image stored in memory, passing any entered arguments to the called application. When booting a Linux kernel, *arg* can be the address of an initrd image. If *addr* is not specified, the environment variable *loadaddr* is used as the default.

Definition: `bootm [addr [arg …]]`

bootp

The **bootp** command boots an image via a network connection using the BootP/TFTP protocol. If *loadaddress* or *bootfilename* is not specified, the environment variables *loadaddr* and *bootfile* are used as the default.

Definition: `bootp [loadAddress] [bootfilename]`

To use network download commands (e.g., **bootp**, **bootvx**, **rarpboot**, **tftpboot**), the environment variables listed in Table 9-4 must be configured. To set a static IP, these environment variables must be specified through the command line interface.

Table 9-4: *Static IP Ethernet Configuration*

Environment Variable:	Description:
ipaddr	Local IP address for the board
serverip	TFTP/NFS server address
netmask	Net mask
gatewayip	Gateway IP address
ethport	eTSEC1 default
ethaddr[1]	MAC address

1. Ensure that each MAC address on the network is unique.

bootv

The **bootv** command checks the checksum on the primary image (in flash) and boots it, if valid. If it is not valid, it checks the checksum on the secondary image (in flash) and boots it, if valid. If neither checksum is valid, the command returns back to the monitor prompt.

Definition: Verify bootup.

```
bootv
```

Write image to flash and update NVRAM.

```
bootv <primary|secondary> write <source> <dest> <size>
```

Update NVRAM based on image already in flash.

```
bootv <primary|secondary> update <source> <size>
```

Check validity of images in flash.

```
bootv <primary|secondary> check
```

bootvx

The **bootvx** command boots VxWorks® from an ELF image, where *address* is the load address of the VxWorks ELF image. To use this command, the environment variables listed in Table 9-4 must be configured.

Definition: `bootvx [ address ]`

dhcp

The **dhcp** command invokes a Dynamic Host Configuration Protocol (DHCP) client to obtain IP and boot parameters by sending out a DHCP request and waiting for a response from a server.

Definition: `dhcp [loadaddress] [bootfilename]`

To use the **dhcp** command, your DHCP server must be configured with the variables designated in Table 9-5.

Table 9-5: *DHCP Ethernet Configuration*

Environment Variable:	Description:	Value[1]:
ipaddr	Local IP address for the board, configured by DHCP	e.g., 192.168.1.1
serverip	TFTP/NFS server address value must be configured after the DHCP IP address is acquired[2]	e.g., 192.168.1.2
netmask	Net mask, obtained by DHCP	–
gatewayip	Gateway IP address, obtained by DHCP	–
ethport	eTSEC1 default	–
ethaddr[3]	MAC address	00:80:F9:xx:xx:xx
autoload[4]	Boot image from TFTP server after DHCP acquisition	no

1. Values for *ethaddr*, *netdev* and *autoload* are set by the user.
2. The value obtained by the DHCP server may not be applicable to your development application.
3. Ensure that each MAC address on the network is unique.
4. If *autoload* is not set or configured to "yes," ensure that the DHCP provides proper information for autoboot. If proper autoboot information is not provided, an error may occur.

rarpboot

The **rarpboot** command boots an image via a network connection using the RARP/TFTP protocol. If *loadaddress* or *bootfilename* is not specified, the environment variables *loadaddr* and *bootfile* are used as the default. To use this command, the environment variables listed in Table 9-4 must be configured.

Definition: `rarpboot [loadaddress] [bootfilename]`

tftpboot

The **tftpboot** command loads an image via a network connection using the TFTP protocol. The environment variable's *ipaddr* and *serverip* are used as additional parameters to this command. If *loadaddress* or *bootfilename* is not specified, the environment variables *loadaddr* and *bootfile* are used as the default. To use this command, the environment variables listed in Table 9-4 must be configured.

The port used is defined by the *ethport* environment variable. If `all` is selected for *ethport*, the TFTP process will cycle through each port until a connection is found or all ports have failed.

Definition: `tftpboot [loadaddress] [bootfilename]`

FILE LOAD COMMANDS

The file load commands load files over the serial port.

loadb

The **loadb** command loads a binary file over the serial port. The command takes two optional parameters:

offset: The address offset parameter allows the file to be stored in a location different than what is indicated within the binary file by adding the value *off* to the file's absolute address.

baudrate: The baudrate parameter allows the file to be loaded at *baud* instead of the monitor's console baudrate.

The file is not automatically executed, the **loadb** command only loads the file into memory.

Definition: `loadb [off] [baud]`

loads

The **loads** command loads an S-Record file over the serial port. The command takes two optional parameters:

offset: The address offset parameter allows the file to be stored in a location different than what is indicated within the S-Record file by adding the value *off* to the file's absolute address.

baudrate: The baudrate parameter allows the file to be loaded at *baud* instead of the monitor's console baudrate.

The file is not automatically executed, the **loads** command only loads the file into memory.

Definition: `loads [off] [baud]`

MEMORY COMMANDS

The memory commands allow you to manipulate specific regions of memory. For some memory commands, the data size is determined by the following flags:

Definition: The flag .b is for data in 8-bit bytes.

Definition: The flag .w is for data in 16-bit words.

Definition: The flag .l is for data in 32-bit long words.

These flags are optional arguments and describe the objects on which the command operates. If you do not specify a flag, memory commands default to 32-bit long words. Numeric arguments are in hexadecimal.

cmp

The **cmp** command compares *count* objects between *addr1* and *addr2*. Any differences are displayed on the console display.

Definition: `cmp [.b, .w, .l] addr1 addr2 count`

cp

The **cp** command copies *count* objects located at the *source* address to the *target* address.

Note: *If the target address is located in the range of the flash device, it will program the flash with count objects from the source address. The cp command does not erase the flash region prior to copying the data. The flash region must be manually erased using the erase command prior to using the cp command.*

Definition: `cp [.b, .w, .l] source target count`

Example: In this example, the **cp** command is used to copy 0x1000, 32-bit values from address 0x100000 to address 0x80000.

```
=> cp 100000 80000 1000
```

find

The **find** command searches from *base_addr* to *top_addr* looking for *pattern*. For the **find** command to work properly, the size of *pattern* must match the size of the object flag. The *-a* option searches for the absence of the specified pattern.

Definition: `find [.b, .w, .l] [-a] base_addr top_addr pattern`

Example: In this example, the **find** command is used to search for the 32-bit pattern 0x12345678 in the address range starting at 0x40000, and ending at 0x80000.

```
=> find.1 40000 80000 12345678

Searching from 0x00040000 to 0x00080000
Match found: data = 0x12345678 Adrs = 0x00050a6c

=>
```

md

The command **md** displays the contents of memory starting at *address*. The number of objects displayed can be defined by an optional third argument, *# of objects*. The memory's numerical value and its ASCII equivalent is displayed.

Definition: `md [.b, .w, .l] address [# of objects]`

Example: In this example, the **md** command is used to display thirty-two 16-bit words starting at the physical address 0x80000.

```
=> md.w 80000 20

00080000: ffff ffff ffff ffff ffff ffff ffff ffff   ................
00080010: ffff ffff ffff ffff ffff ffff ffff ffff   ................
00080020: ffff ffff ffff ffff ffff ffff ffff ffff   ................
```

```
00080030: ffff ffff ffff ffff ffff ffff ffff ffff   ................
```

mm

The **mm** command modifies memory one object at a time. Once started, the command line prompts for a new value at the starting address. After a new value is entered, pressing ENTER auto-increments the address to the next location. Pressing ENTER without entering a new value leaves the original value for that address unchanged. To exit the **mm** command, enter a non-valid hexadecimal value (such as x) followed by ENTER.

Definition: `mm [.b, .w, .l] address`

Example: In this example, the **mm** command is used to write random 8-bit data starting at the physical address 0x80000.

```
=> mm.b 80000
00080000: ff ? 12
00080001: ff ? 23
00080002: ff ? 34
00080003: ff ? 45
00080004: ff ?
00080005: ff ? x
=> md.b 80000 6
00080000: 12 23 34 45 ff ff   .#4E
=>
```

nm

The **nm** command modifies a single object repeatedly. Once started, the command line prompts for a new value at the selected address. After a new value is entered, pressing ENTER modifies the value in memory and then the new value is displayed. The command line then prompts for a new value to be written at the same address. Pressing ENTER without entering a new value leaves the original value unchanged. To exit the **nm** command, enter a non-valid hexadecimal value (such as x) followed by ENTER.

Definition: `nm [.b, .w, .l] address`

mw

The command **mw** writes *value* to memory starting at *address*. The number of objects modified can be defined by an optional fourth argument, *count*.

Definition: `mw [.b, .w, .l] address value [count]`

Example: In this example, the **mw** command is used to write the value 0xabba three times starting at the physical address 0x80000.

```
=> mw.w 80000 abba 3
=> md 80000

00080000: abbaabba abbaffff ffffffff ffffffff   ................
00080010: ffffffff ffffffff ffffffff ffffffff   ................
```

```
00080020: ffffffff ffffffff ffffffff ffffffff   ................
00080030: ffffffff ffffffff ffffffff ffffffff   ................
00080040: ffffffff ffffffff ffffffff ffffffff   ................
00080050: ffffffff ffffffff ffffffff ffffffff   ................
00080060: ffffffff ffffffff ffffffff ffffffff   ................
00080070: ffffffff ffffffff ffffffff ffffffff   ................
```

FLASH COMMANDS

The flash commands affect the StrataFlash devices on the ATCA-9305 circuit board. There are four flash banks on the ATCA-9305 board (see "Flash" on page 4-7). They can be accessed by:

- the individual bank (1, 2, 3 or 4), or
- the address, where both banks are regarded as one contiguous address space

The following flash commands access the individual flash bank as flash bank *1*. To access the individual sectors within each flash bank, the sector numbers start at *0* and end at one less than the total number of sectors in the bank. For a flash bank with 128 sectors, the following flash commands access the individual sectors as *0* through *127*.

cp

The **cp** command can be used to copy data into the flash device. For the **cp** command syntax, refer to "Memory Commands" on page 9-11.

erase

The **erase** command erases the specified area of flash memory.

Definition: Erase all of the sectors in the address range from *start* to *end*.

```
erase start end
```

Erase all of the sectors *SF* (first sector) to *SL* (last sector) in flash bank # *N*.

```
erase N:SF[-SL]
```

Erase all of the sectors in flash bank # *N*.

```
erase bank N
```

Erase all of the sectors in all of the flash banks.

```
erase all
```

flinfo

The **flinfo** command prints out the flash device's manufacturer, part number, size, number of sectors, and starting address of each sector.

Definition: Print information for all flash memory banks.

```
flinfo
```

Print information for the flash memory in bank # *N*.

```
flinfo N
```

protect

The **protect** command enables or disables the flash sector protection for the specified flash sector. Protection is implemented using software only. The protection mechanism inside the physical flash part is not being used.

Definition: Protect all of the flash sectors in the address range from *start* to *end*.

```
protect on start end
```

Protect all of the sectors *SF* (first sector) to *SL* (last sector) in flash bank # *N*.

```
protect on N:SF[-SL]
```

Protect all of the sectors in flash bank # *N*.

```
protect on bank N
```

Protect all of the sectors in all of the flash banks.

```
protect on all
```

Remove protection on all of the flash sectors in the address range from *start* to *end*.

```
protect off start end
```

Remove protection on all of the sectors *SF* (first sector) to *SL* (last sector) in flash bank # *N*.

```
protect off N:SF[-SL]
```

Remove protection on all of the sectors in flash bank # *N*.

```
protect off bank N
```

Remove protection on all of the sectors in all of the flash banks.

```
protect off all
```

EEPROM/I^{2}C COMMANDS

This section describes commands that allow you to read and write memory on the serial EEPROMs and I^{2}C devices.

eeprom

The **eeprom** command reads and writes from the EEPROM. For example:

```
eeprom read 53 100000 1800 100
```

reads 100 bytes from offset 0x1800 in serial EEPROM 0x53 (right-shifted 7-bit address) and places it in memory at address 0x100000.

Definition: Read/write *cnt* bytes from *devaddr* EEPROM at offset *off*.

```
eeprom read devaddr addr off cnt
eeprom write devaddr addr off cnt
```

icrc32

The **icrc32** computes a CRC32 checksum.

Definition:
```
icrc32 chip address[.0, .1, .2] count
```

iloop

The **iloop** command reads in an infinite loop on the specified address range.

Definition:
```
iloop chip address[.0, .1, .2] [# of objects]
```

imd

The **imd** command displays the primary I^2C bus memory. For example:

```
imd 53 1800.2 100
```

displays 100 bytes from offset 0x1800 of I^2C device 0x53 (right-shifted 7-bit address). The `.2` at the end of the offset is the length, in bytes, of the offset information sent to the device. The serial EEPROMs all have two-byte offset lengths. The Real-Time Clock (RTC) has a one-byte offset length. The temperature sensors have zero-byte offset lengths.

Definition:
```
imd chip address[.0, .1, .2] [# of objects]
```

imm

The **imm** command modifies the primary I^2C memory and automatically increments the address.

Definition:
```
imm chip address[.0, .1, .2]
```

imw

The **imw** command writes (fills) memory.

Definition:
```
imw chip address[.0, .1, .2] value [count]
```

inm

The **inm** command modifies I^2C memory, reads it, and keeps the address.

Definition:
```
inm chip address[.0, .1, .2]
```

iprobe

The **iprobe** command probes to discover valid primary I^2C bus chip addresses.

Definition:

```
iprobe
```

IPMC COMMANDS

IPMI Baseboard Management Controller (BMC) watchdog is supported and serviced throughout the monitor boot process. The BMC watchdog is disabled if the monitor goes to the monitor prompt.

bootdev

The **bootdev** command gets or sets the initial boot bank. Get prints out the flash bank set as initial boot device.

Definition:

```
bootdev get
```

The IPMC sets the hardware strapping for the initial boot device.

```
bootdev set <bank>
```

Where <bank> is either b0 or b1 for the corresponding flash bank, or b3 to boot from socket and if a shunt is installed on J9 [1:2].

fru

The **fru** command opens, closes, saves, sets, shows, dumps, and loads *fru* data to and from the IPMC.

Definition:

```
fru <command> [ arg1 arg2 … ]
command := [ open | close | save | set | show | dump | load | create ]
fru open <id>
fru close
fru save
fru set <section [chassis|board|product]><field><value>

fru set <section> <field> <value>
section := [ chassis | board | product ]

fru set chassis <field> <value>
field := [ type | part | serial ]

fru set board <field> <value>
field := [ date | maker | name | serial | part | file ]

fru set product <field> <value>
field := [ maker | name | part | version |serial | asset | file ]

fru show
fru dump <address>
fru load <address><size>
```

Set data in the internal use area.

```
fru set internal <source addr> <internal use offset> <count>
```

The **fru create** command loads a default *fru* image to a blank *fru* device.

```
fru create <id> default <product name>
fru create <id> <address> <size> <product name>
```

fruinit

The **fruinit** command initializes the following *fru* data fields: part number, build date, and serial number in the board and product sections.

Definition:

```
fruinit <fru id> <part number> <build date> [ serial number ]
```

fruled

The **fruled** command allows the application programmer to get the status of the red out-of-service LED or to turn the LED on or off when an application fails to load.

Definition:

```
fruled get <fru id> <led id> <led state> <led function (on/off)> <on time> <color>
fruled set <fru id> <led id> <led function (on/off)> <on time> <color>
```

Example: Turns the red out-of-service LED on.

```
fruled set 0 1 0xff 0 2
```

Turns the red out-of-service LED off.

```
fruled set 0 1 0 0 2
```

ipmchpmfw

The **ipmchpmfw** command restores the previous IPMC firmware from the backup IPMC firmware stored in the controller. The *upgrade* argument upgrades the IPMC firmware with the upgrade image held in memory.

Definition:

```
ipmchpmfw [restore] [upgrade <source address>]
```

sensor

The **sensor** command probes, reads, and prints the sensor information from the IPMI.

Definition:

```
sensor [probe|read|dump]
```

Sensor probe prints out each sensor number and name.

```
sensor probe <sensor number>
```

Sensor read prints out the sensor reading for sensor.

```
sensor read <sensor number>
```

Sensor dump prints out the raw Sensor Data Record (SDR) information for sensor.

```
sensor dump <sensor number>
```

ENVIRONMENT PARAMETER COMMANDS

The monitor uses on-board, non-volatile memory for the storage of environment parameters. Environment parameters are stored as ASCII strings with the following format.

```
<Parameter Name>=<Parameter Value>
```

Some environment variables are used for board configuration and identification by the monitor. The environment parameter commands deal with the reading and writing of these parameters. Refer to “MPC8548 Environment Variables” on page 9-26 for a list of monitor environment variables.

Redundant environment parameters allow you to store a “backup” copy of environment parameters should they ever become corrupt. The redundant environment parameters are only used if the main parameters are corrupt.

To save environment variables:

1. Use **moninit** to save default environment variables to both primary and secondary environment parameters.
2. Use **saveenv** to save to the primary environment variables.
3. Set the next save to the secondary image.

printenv

The **printenv** command displays all of the environment variables and their current values to the display.

Definition: Print the values of all environment variables.

```
printenv
```

Print the values of all environment variable (exact match) ‘name’.

```
printenv name …
```

saveenv

The **saveenv** command writes the environment variables to non-volatile memory.

Definition:

```
saveenv
```

setenv

The **setenv** command adds new environment variables, sets the values of existing environment variables, and deletes unwanted environment variables.

Definition: Set the environment variable *name* to *value* or adds the new variable *name* and *value* to the environment.

```
setenv name value
```

Removes the environment variable *name* from the environment.

```
setenv name
```

TEST COMMANDS

The commands described in this section perform diagnostic and memory tests.

diags

The **diags** command runs the Power-on Self-test (POST).

Definition:

```
diags
```

mtest

The **mtest** command performs a simple SDRAM read/write test.

Definition:

```
mtest [start [end [pattern]]]
```

um

The **um** command is a destructive memory test. Press the 'q' key to quit this test; the monitor completes running the most recent iteration, and exits to the default prompt after displaying cumulative results for the completed iterations.

Definition:

```
um [.b, .w, .l] base_addr [top_addr]
```

OTHER COMMANDS

This section describes all the remaining commands supported by the ATCA-9305 monitor.

autoscr

The **autoscr** command runs a script, starting at address *addr*, from memory.
A valid **autoscr** header must be present.

Definition:

```
autoscr [addr]
```

base

The **base** command prints or sets the address offset for memory commands.

Definition: Displays the address offset for the memory commands.

```
base
```

Sets the address offset for the memory commands to *off*.

```
base off
```

bdinfo

The **bdinfo** command displays the Board Information Structure.

Definition: `bdinfo`

coninfo

The **coninfo** command displays the information for all available console devices.

Definition: `coninfo`

crc32

The **crc32** command computes a CRC32 checksum on *count* bytes starting at *address*.

Definition: `crc32 address count`

date

The **date** command will set or get the date and time, and reset the RTC device.

Definition: Set the date and time.

```
date [MMDDhhmm[[CC]YY][.ss]]
```

Display the date and time.

```
date
```

Reset the RTC device.

```
date reset
```

echo

The **echo** command echoes *args* to console.

Definition: `echo [args..]`

enumpci

The **enumpci** command enumerates the PCI bus (when the hardware is the PCI Root Complex in the system).

Definition: `enumpci`

go

The **go** command runs an application at address *addr*, passing the optional argument *arg* to the called application.

Definition: *go addr* [arg…]

help

The **help** (or **?**) command displays the online help. Without arguments, all commands are displayed with a short usage message for each. To obtain more detailed information for a specific command, enter the desired command as an argument.

Definition: `help [command …]`

iminfo

The **iminfo** command displays the header information for an application image that is loaded into memory at address *addr*. Verification of the image contents (magic number, header, and payload checksums) are also performed.

Definition: `iminfo addr [addr …]`

isdram

The **isdram** command displays the SDRAM configuration information (valid chip values range from 50 to 57).

Definition: `isdram addr`

loop

The **loop** command executes an infinite loop on address range.

Definition: `loop [.b, .w, .l] address number_of_objects`

memmap

The **memmap** command displays the board's memory map layout.

Definition: `memmap`

moninit

The **moninit** command resets the NVRAM and serial number, and writes the monitor to flash. The ATCA-9305 must be booted from the boot socket for this command to function in the default state. The proper region of flash memory will be unlocked and erased prior to copying the monitor software into it.

The command flags, .1 or .2, force the monitor to be programmed to a single (.1) bank of flash or dual (.2) banks of flash. If the command flags are not used, then **moninit** checks for the number of banks of flash. If there are two banks of flash, then **moninit** automatically programs both banks for redundancy. Also, the serial number can be obtained from the fru data if "fru" is used as a parameter.

Definition: Initialize environment variables and serial number in NVRAM and copy the monitor from the socket to NOR (soldered) flash.

```
moninit[.1, .2] <serial# or "fru">
```

Initialize environment variables and serial number in NVRAM but do not update the monitor in NOR flash.

```
moninit[.1, .2] <serial# or "fru"> noburn
```

Initialize environment variables and serial number in NVRAM and copy the monitor from <*src_address*> into NOR flash.

```
moninit[.1, .2] <serial# or "fru"> <src_address>
```

pci

The **pci** command enumerates the PCI bus. It displays enumeration information about each detected device. The **pci** command allows you to display values for and access the PCI Configuration Space.

Definition: Display a short or *long* list of PCI devices on the bus specified by *bus*.

```
pci [bus] [long]
```

Show the header of PCI device *bus.device.function*.

```
pci header b.d.f
```

Display the PCI configuration space (CFG).

```
pci display[.b, .w, .l] b.d.f [address] [# of objects]
```

Modify, read, and keep the CFG address.

```
pci next[.b, .w, .l] b.d.f address
```

Modify, automatically increment the CFG address.

```
pci modify[.b, .w, .l] b.d.f address
```

Write to the CFG address.

```
pci write[.b, .w, .l] b.d.f address value
```

phy

The **phy** command reads or writes to the contents of the PHY registers. The values changed via this command are not persistent and clear after a hard or soft reset. The port options are all, eTSEC1, eTSEC2, and base1 and base2 via the switch. “R” reads the register contents at the address specified. “W” writes the address value to the register address specified. “A” reads the contents of all registers.

Definition: `phy [port] [R|W|A] (address) (value)`

Example: The following is an example of a read from register address 0x1a.

```
phy eTSEC2 r 0x1a
```

The following is an example of a write to register address 0x1a where 0 is the data to write.

```
phy eTSEC2 w 0x1a 0
```

ping

The **ping** command sends a ping over Ethernet to check if the host can be reached. The port used is defined by the *ethport* environment variable. If `all` is selected for *ethport*, the ping process cycles through each port until a connection is found or all ports have failed.

Definition:

```
ping host
```

reset

The **reset** command performs a hard reset of the CPU by writing to the reset register on the board. Without any arguments, the ATCA-9305 CPU is reset.

Definition:

```
reset
```

run

The **run** command runs the commands in an environment variable *var*.

Definition:

```
run var [ … ]
```

Use $ for variable substitution; the syntax "$(*variable_name*)" should be used for variable expansion.

Example:

```
=> setenv cons_opts console=tty0 console=ttyS0,\$(baudrate)
=> printenv cons_opts cons_opts=console=tty0 console=ttyS0,$(baudrate)
```

Use the \ character to escape execution of the $ as seen in the **setenv** command above. In this example, the value for baudrate will be inserted when cons_opts is executed.

script

The **script** command runs a list of monitor commands out of memory. The list is an ASCII string of commands separated by the `;` character and terminated with the `;;` characters. <*script address*> is the starting location of the script.

Note: *A script is limited to 1000 characters.*

Definition:

```
script <script address>
```

showmac

The **showmac** command displays the Processor MAC addresses assigned to each Ethernet port.

Definition:

```
showmac
```

showpci

The **showpci** command scans the PCI bus and lists the base address of the devices.

Definition: `showpci`

sleep

The **sleep** command executes a delay of *N* seconds.

Definition: Delay execution for N seconds (N is a decimal value).

```
sleep N
```

switch_reg

The **switch_reg** command reads or writes to the Ethernet core switch registers. The values changed via this command are not persistent and clear after a hard or soft reset. Option values are as follows: switch (core or fp), port (0 - 25), block (1-7), and sub-block (0-15). “R” reads the register contents at the address specified. “W” writes the address value to the register address specified.

Definition: `switch_reg [switch] [port op | block sub-block op [R|W]] (address) (value)`

Example: The following is an example of a read of register address 0x1a.

```
switch_reg core 0 r 0x1a
```

The following is an example of a write to register address 0x1a where 0 is the data to write.

```
switch_reg core 0 w 0x1a 0
```

version

The **version** command displays the monitor’s current version number.

Definition: `version`

vlan

The **vlan** command creates one or more new VLANs using *vid* as the VLAN identification (VID) value and deletes one or more existing VLANs whose VLAN ID matches the VLAN ID value *vid*. These variables are set using a comma-separated list of port names. This command sets an untagged port-based VLAN and the VLAN table entry with the port’s default VID. In this configuration, each port is assigned to one VLAN.

Definition:

```
vlan add <vid1>=portlist1> <vid2>=<portlist2>…
vlan delete <vid1> <vid2>…
vlan show
```

Example: To create VLAN 1 on the core switch:

```
vlan add 1=14,15
```

To delete VLAN 1 on the core switch:

```
vlan delete 1
```

MPC8548 ENVIRONMENT VARIABLES

Press the 's' key on the keyboard during reset to force the default monitor environment variables to be loaded during hardware initialization but before diagnostic testing.

Table 9-6: *Standard Environment Variables*

Variable:	Default Value:	Description:
baudrate	115200	Console port baud rate Valid rates: 9600, 14400, 19200, 38400, 57600, 115200
bmc_wd_timeout	-1	This sets the time-out in seconds for the BMC watchdog before booting the OS. If set to -1, then the BMC watchdog is disabled before booting the OS. Valid options: -1, 1-65535
bootcmd	" "	Command to execute when auto-booting or executing the 'bootd' command
bootdelay	1	Choose the number of seconds the Monitor counts down before booting user application code Valid options: time in seconds, -1 to disable autoboot
bootfile	" "	Path to boot file on server (used with TFTP)–set this to "path/file.bin" to specify filename and location of the file to load.
bootretry	-1	Set the number of seconds the Monitor counts down before booting user application code (used only with autoboot). If the boot commands fails, it will try again after *bootretry* seconds. Valid options: time in seconds, -1 to disable *bootretry*
bootstopkey	h	Press during power-up/reset initialization to terminate the monitor autoboot sequence and go to the monitor prompt.
clearmem	on	Select whether to clear unused SDRAM (memory used by monitor is excluded) on power-up and reset. Valid options: on, off
dcache	on	Enables the processor L1 data cache Valid options: on, off
ecc	on	Enable ECC initialization–all of memory is cleared during ECC initialization. Valid options: on, off
ecc_1bit_report	off	Select the reporting of single bit, correctable ECC errors to the console (errors of 2 or more bits are always reported) Valid options: on, off
ethaddr	00:80:F9:97:00:00-00:80:F9:97:FF:FF	ATCA-9305 board Ethernet address for TSEC_1 port, the last digits are the board serial number in hex.

Variable:	Default Value:	Description: (continued)
eth1addr	00:80:F9:98:00:00-00:80:F9:98:FF:FF	ATCA-9305 board Ethernet address for TSEC_2 port, the last digits are the board serial number in hex.
fru_id	undefined	Corresponds to ATCA-9305 processing resources Valid options: Not defined in default configuration–reported at bootup from the IPMC
gatewayip	0.0.0.0	Select the network gateway machine IP address
hostname	EMERSON_ATCA-9305	Target hostname
icache	on	Enables the processor L1 instruction cache Valid options: on, off
ipaddr	0.0.0.0	Board IP address
l2cache	on	Enables the L2 cache Valid options: on, off
loadaddr	0x100000	Define the address to download user application code (used with TFTP)
model	ATCA-9305	Board model number
ncip	undefined	Sets the IP address and the destination port, format is <ip_addr>;<port>
netmask	0.0.0.0	Board sub-network mask
powerondiags	on	Turns POST diagnostics on or off after power-on/reset Valid options: on, off
preboot	undefined	Command to execute immediately before starting the CONFIG_BOOTDELAY countdown and/or running the auto-boot command entering the interactive mode
rootpath	eng/emerson/	Path name of the NFS' server root file system
serial#	xxxxx	Board serial number
serverip	0.0.0.0	Boot server IP address
tftp_port	eTSEC_1	Selects which Ethernet port will be used for tftp Valid options: eTSEC_1, eTSEC_2

The monitor supports optional environment variables that enable additional functionality. The **moninit** command (see "moninit" on page 9-22) clears all environment variables and sets the standard environment variables to the default values. All optional environment variables are removed after **moninit**. However, it can clear all optional variables.

Table 9-7: *Optional Environment Variables*

Variable[1]:	Description:
app_lock_base	Assigns where to start block lock protection at the base of NOR (soldered) flash. If assigned region does not fall within the NOR flash area, no user/application locking will occur, except for the monitor block-locking protection.
app_lock_size	Size of user NOR (soldered) flash protection area.
bootverifycmd	Sets the U-Boot boot command that is used to execute the primary and secondary application images when using the **bootv** command. If not defined, **bootv** uses the U-Boot **go** command as the default.
carrier_num	This is a slot within a shelf defined by the zone 1 hardware address corresponding to the logical slot address.
e_keying	Determines whether switch ports should be configured.
pci_memsize	Sets the amount of SDRAM memory made available on the PCI bus. The minimum setting is 16 megabytes. If not set, 128 MB of SDRAM are available over PCI. This parameter takes a hex value. Valid options: all, size in hex (0x8000000=128 MB)
pram	This memory region is at the very top of memory and can be reserved–not to be cleared on start-up or reset. Default size of the protected memory region is 0. *pram* is defined in kilobytes and is a base 10 number. The smallest allowable size is 4 (4 KB) and the largest recommended size is 32768 (32 MB). *pram* should be 4 KB aligned, otherwise U-Boot will round *pram* to the next 4 KB size.
sec_bootargs	Sets the boot arguments that are passed into the secondary application images when using the **bootv** command. If not defined, **bootv** will pass the *bootargs* configuration parameters into both the primary and secondary application images.
shelf_addr	ATCA chassis shelf address provided by shelf-manager Not defined in default configuration–reported at bootup from the IPMC

1. The **moninit** command does not initialize these variables. Each parameter is only defined if a change from the default setting is desired and is not defined after initialization of the environment variables.

TROUBLESHOOTING

To bypass the full board initialization sequence, attach a terminal to the console located on the front of the ATCA-9305. Configure the terminal parameters to be:

9600 bps, no parity, 8 data bits, 1 stop bit

Reset the ATCA-9305 while holding down the ‘s’ key. Pressing the ‘s’ key forces a configuration based on default environment variables.

DOWNLOAD FORMATS

The ATCA-9305 monitor supports binary and Motorola® S-Record download formats, as described in the following sections.

Binary

The binary formats (and associated commands) include:

- Executable binary files (**go**)
- VxWorks and QNX® ELF (**bootm**, **bootvx**, or **bootelf**)
- Compressed (gzipped) VxWorks and QNX ELF (**bootm**)
- Linux kernel images (**bootm**)
- Compressed (gzipped) Linux kernel images (**bootm**)

Motorola S-Record

S-Record download uses the standard Motorola S-Record format. This includes load address, section size, and checksum all embedded in an ASCII file.

(blank page)

Acronyms

AMC	Advanced Mezzanine Card
ASCII	American Standard Code for Information Interchange
ATCA	Advanced Telecom Computing Architecture or AdvancedTCA
BMC	Baseboard Management Controller
CIO	Common I/O (RLDRAM)
Cmd	Command code
CPU	Central Processing Unit
CRC	Cyclic Redundancy Code
CSA	Canadian Standards Association
DDR	Double Data Rate
EC	European Community
ECC	Error-correcting Code
EIA	Electronic Industries Alliance
EMC	Electromagnetic Compatibility
ESD	Electrostatic Discharge
ETSI	European Telecommunications Standards Institute
EXP	Extreme Processor
FCC	Federal Communications Commission
FRU	Field Replaceable Unit
GbE	Gigabit Ethernet
GNU	GNU's Not Unix
GPL	General Public License
I^2C	Inter-integrated Circuit
IEC	International Electrotechnical Commission
IPMB	Intelligent Platform Management Bus
IPMI	Intelligent Platform Management Interface
ISP	In-system Programmable
ITP	In-target Probe
JTAG	Joint Test Action Group
KCS	Keyboard Controller Style
LED	Light-emitting Diode
LPC	Low Pin Count
LUN	Logical Unit Number
MAC	Medium/media Access Control/controller
NEBS	Network Equipment-Building System
netFn	Network Function Code
NSP	Network Services Processor

Acronyms: (continued)

OEM	Original Equipment Manufacturer
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PHY	Physical Interface
PLD	Programmable Logic Device
POST	Power-on Self Test
RLDRAM	Reduced Latency Dynamic Random Access Memory
RMA	Return Merchandise Authorization
SCP	Secure Communications Processor
SDR	Sensor Data Record
SDRAM	Synchronous Dynamic Random Access Memory
SEL	System Event Log
SERDES	Serializer/deserializer
SIO	Separate I/O (RLDRAM)
SO-CDIMM	Small-outline Clocked Dual In-line Memory
SPI-4.2	System Packet Interface level 4 phase 2
SROM	Serial Read Only Memory
TBD	To Be Determined
UART	Universal Asynchronous Receiver/transmitter
UL	Underwriters Laboratories
USB	Universal Serial Bus
VLP	Very Low Profile
XAUI	10 Gigabit Attachment Unit Interface

Index

Index (continued)

N

P

R

S

T

(blank page)

Notes

Emerson Network Power
The global leader in enabling
Business-Critical Continuity™

AC Power Systems
Connectivity
DC Power Systems
Embedded Computing
Embedded Power
Integrated Cabinet Solutions
Outside Plant
Power Switching & Controls
Precision Cooling
Services
Site Monitoring
Surge & Signal Protection

Emerson Network Power
Offices: Tempe, AZ U.S.A. 1 800 759 1107 or +1 602 438 5720
Madison, WI U.S.A. 1 800 356 9602 or +1 608 831 5500
Shanghai, China +8610 85631122 Paris, France +33 1 60 92 31 20 Tokyo, Japan +81 3 5403 2730
Munich, Germany +49 89 9608 2333 Hong Kong, China +852 2176 3540 Tel Aviv, Israel +972 3 568 4387
www.EmersonNetworkPower.com/EmbeddedComputing

EMERSON. CONSIDER IT SOLVED.™